Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2010

João Paulo Ligeiro Feteira Parracho

Gateway LinuxMCE - OpenWrt para Aplicações Domóticas

João Paulo Ligeiro Feteira Parracho

Gateway LinuxMCE - OpenWrt para Aplicações Domóticas

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e de Telecomunicações, realizada sob a orientação científica do Doutor Arnaldo Silva Rodrigues de Oliveira, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor Nuno Miguel Gonçalves Borges de Carvalho
Professor Associado com Agregação da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor José Carlos Meireles Monteiro Metrôlho
Professor Adjunto do Departamento de Engenharia Informática da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

Prof. Doutor Arnaldo Silva Rodrigues de Oliveira
Professor Auxiliar da Universidade de Aveiro (Orientador)

agradecimentos / acknowledgements

Gostaria de agradecer, em primeiro lugar, ao Prof. Doutor Arnaldo Oliveira, orientador deste projecto, pelo tempo, dedicação e excelência demonstrados no decorrer de todo o trabalho e pelo apoio disponibilizado na sua concretização.

Não posso deixar de prestar, ainda, o devido agradecimento a todos os professores que me instruíram ao longo destes cinco anos e que foram, sem dúvida, decisivos e fundamentais na minha formação, tanto a nível pessoal, como a nível académico. Quero também deixar, aqui, uma nota de apreço a todos os colegas e amigos com quem partilhei a formação académica e, em particular, ao meu grande amigo Paulo, aos meus colegas de sala, Anabela, Ricardo, Carlos e Tiago que, neste último ano decisivo, estiveram sempre dispostos a ajudar e a partilhar ideias e conhecimento.

Por fim, mas não menos importante, uma palavra de apreço especial a todos os meus familiares, em particular à minha Mãe, que nunca deixou de acreditar em mim, e sempre me incentivou a continuar, mesmo nas situações mais adversas e à Ana por toda a paciência, apoio, carinho que me dedicou e me dedica todos os dias.

Palavras Chave Domótica, Casa do Futuro, Casa Inteligente, *LinuxMCE*

Resumo

A domótica é um campo da automação em franca expansão e, hoje em dia, é recorrente vê-la incorporada em novas construções habitacionais que pretendem níveis de conforto, segurança e tecnologia acima da média. No entanto, um dos problemas desta nova tendência tecnológica, tal como a maior parte delas, reside no seu custo muito elevado.

Assim, com esta dissertação pretende implementar-se um sistema de baixo custo, devido ao uso de sensores e ao apoio numa distribuição *open source* dedicada a esta área, o *LinuxMCE*.

Como primeira abordagem, foi estudada a plataforma em questão, métodos da sua instalação e compilação e integração de novos sensores. Depois desta fase inicial, criou-se em C/C++ uma ponte de comunicação entre eles e o sistema (a *gateway*), baseada em *sockets* TCP/IP e, por fim, um demonstrador.

Quanto a resultados obtidos, conseguiu-se instalar e configurar a distribuição referida, estabelecer a conexão entre os vários sensores e o *LinuxMCE* e assegurar o funcionamento do sistema a nível geral (nomeadamente a nível temporal). Existem, no entanto, algumas limitações, visto este sistema já implementar um esquema de troca de mensagens entre ele e os dispositivos externos, que não foi possível adaptar aos sensores.

Key Words Domotics, House of the Future, Smart Home, *LinuxMCE*

Abstract

Domotics is a field of automation in fast growing expansion, and nowadays it's usual to notice it incorporated in new habitacional constructions that aim for unprecedented levels of comfort, security and technology. Despite that, one of the issues with this new technological trend, as with most of them, is related with its high cost.

Therefore, the goal of this dissertation, is to implement a low cost system of this type, due to the usage of sensors and *LinuxMCE*, a support open source software dedicated to this area.

As a first approach, one has studied the platform, his methods of installation and how to compile and integrate the new sensors. Afterwards, it was created a bridge in C/C++ based on TCP/IP sockets to accommodate the communication between them and the system and a showcase of the overall work.

Regarding final results, one has accomplished to properly install and configure the distribution, establish the connection among the various sensors and *LinuxMCE* and having this last one working (regarding temporal restrains). There are however some limitations, because this system already implemented a scheme of messaging exchange from and to the external devices, that was not possible to adapt to the sensors.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Enquadramento

A domótica é um conceito que visa automatizar e tornar mais eficazes as funções de uma casa que, normalmente, seriam efectuadas de forma manual, desperdiçando tempo, factor de extrema importância face ao ritmo da vida actual. A palavra domótica vem da junção das palavras *domus/domo*[1] e robótica (controlo automatizado de algo) [1].

Esta tecnologia está direccionada para a melhoria da qualidade de vida, nomeadamente, nas áreas de segurança, conforto e gestão técnica. É, por isso, um sistema completo e não apenas um sistema remoto. Com esta tecnologia, muitas das actividades do dia-a-dia passam a ser feitas automaticamente porque são previamente programadas, mantendo, no entanto, o controlo manual de todo o sistema, cujas definições podem ser alteradas, se assim o desejarmos.

A domótica pode reunir iluminação, entretenimento, segurança, telecomunicações e climatização num só sistema. Permite fazer da nossa casa um parceiro crucial na gestão das tarefas quotidianas e, em parte, da vida em si. A habitação deixa de ser uma estrutura passiva tornando-se em vez disso, numa ferramenta que ajuda a desfrutar ao máximo do nosso tempo, aumentando a segurança e até mesmo poupando energia.

Ela envolve diferentes áreas, requerendo um vasto conjunto de conhecimentos para a fazer funcionar. Num sistema comum, um controlador central recebe sinais de dispositivos controladores e depois reencaminha os mesmos para as aplicações e sistemas distribuídos pela casa. O servidor central tem o papel de gerir e encaminhar as comunicações por toda a casa. Como utilizador é possível interagir com o sistema através de teclados, *touchscreens*, ecrãs de TV, computadores, telefones, comandos remotos ou outros dispositivos.

Hoje em dia, a domótica assenta sobre vários protocolos. Entre os mais importantes, podem destacar-se o *Insteon*, KNX, *LonWorks*, X10, *Z-Wave* e *Zigbee*.

Os produtos associados à domótica geralmente são modulares, o que significa que se podem facilmente adicionar ou remover funções ao sistema com pouco ou nenhum efeito no funcionamento dos outros produtos associados.

[1] casa (latim)

Todas as novas habitações estão a começar a adoptar a domótica e a possuir sistemas mais ou menos completos de automação. Esta necessidade surge de uma procura, por parte das pessoas, em adquirirem casas mais inteligentes, seguras, confortáveis e eficientes energeticamente e, por conseguinte, em resposta a uma evolução da sociedade, que está cada vez mais dependente da tecnologia, dentro e fora de casa.

As empresas que trabalham, neste domínio, têm ainda um mercado reduzido, mas com grandes possibilidades de expansão e, tendo em conta, o facto de a instalação ser feita, muitas vezes, à medida para cada cliente e os factores anteriormente mencionados, é previsível que esta área venha a ter um grande potencial económico e de desenvolvimento. Devido aos elevados requisitos a nível de instalação, manutenção, actualização e, consequentemente, monetários, a expansão deste tipo de serviços e produtos ainda não atinge uma grande percentagem das casas em território nacional. Apesar da existência de um número considerável de normas (tanto privadas como públicas), ainda não há um *standard* definido, nem tão pouco consenso, relativamente ao uso de um tipo de cablagem específico para as variadas vertentes que podem ser contempladas numa instalação deste tipo e que ajudariam, sem dúvida, a baixar os custos do produto final e a permitir uma melhor integração entre todos os subsistemas da habitação.

No futuro, pensaremos como foi possível viver sem o contributo de, por exemplo, sistemas de segurança e de multimédia avançados, divisões atentas ao gosto dos utilizadores nelas presentes, tarefas banais automatizadas entre toda uma miríade de tecnologias e inovações destinadas a proporcionar, no fundo, uma melhor qualidade de vida.

1.2 Motivação

Como já foi referido, os elevados custos e a não uniformização presentes neste segmento de mercado foram preponderantes para a motivação deste trabalho de dissertação. Com a quantidade de redes já presentes nas casas, tanto cabladas como não cabladas, surge uma excelente oportunidade para implementar um sistema de domótica de menor custo do que os existentes actualmente.

Sendo assim, a abordagem a explorar no âmbito destes projectos baseia-se na extensão de um AP*(Access Point)* 802.11x (com interface *Fast/Gigabit Ethernet*) de maneira a incorporar os dispositivos e mecanismos de suporte à domótica e a sua conexão à rede de dados cablada e universal. O AP é controlado por um software aberto e flexível, o *OpenWrt*, que permite assim executar a associação e integração com uma plataforma de gestão domótica, o *LinuxMCE*, também *open source*, de modo transparente e baseando-se no mesmo tipo de comunicações, via *sockets*.

Com esta opção é possível atingir uma modularidade e escalabilidade enormes, podendo ser adicionados vários APs (um para cada divisão por exemplo) conectados entre si pela rede cablada de alto débito e assim expandir o alcance/cobertura a toda a casa, garantindo também uma elevada largura de banda.

1.3 Objectivos

Neste projecto pretende-se, então, integrar a interacção com sensores, isto é visualizar o seu estado, numa distribuição *open source* destinada especificamente a este uso, o *LinuxMCE* e, de seguida, executar um demonstrador do sistema.

Para isso, em primeiro lugar, é necessário compreender o sistema a nível de instalação e configuração. Numa segunda fase, perceber como adicionar novos dispositivos ao *LinuxMCE* e aplicar conhecimentos a nível de *sockets* em C/C++ para estabelecer a conexão entre os dois, constituindo assim a *gateway* necessária. Por fim, aplicar o sistema no demonstrador, de modo a ilustrar o funcionamento do mesmo.

1.4 Estrutura da Dissertação

A partir da introdução, esta dissertação está estruturada da seguinte maneira:

- Capítulo 2 - **Estado da Arte** - Neste capítulo é feita uma abordagem aos protocolos mais usados na área da domótica, bem como uma análise detalhada do *LinuxMCE* e uma apresentação do *OpenWrt*.
- Capítulo 3 - **Gateway LinuxMCE - OpenWrt** - Aqui é feita uma apresentação da instalação e configuração do *LinuxMCE*, da criação de *templates* para os dispositivos a adicionar, bem como a sua compilação. É referido ainda como foi implementada a conexão entre os sensores a correr em *OpenWrt* e o sistema.
- Capítulo 4 - **Demonstrador** - Este capítulo trata de apresentar como foi montado e executado o demonstrador, o funcionamento do mesmo e alguns testes. Explica ainda como interagir com o mesmo, através da *Web GUI* (*Graphical User Interface*) e de *Orbiters*[2] Por fim são apresentados também resultados do demonstrador a nível temporal.
- Capítulo 5 - **Conclusões** - Por fim, neste capítulo, referem-se os objectivos iniciais que foram cumpridos, um resumo do trabalho realizado e o que pode ser melhorado e adicionado em iterações futuras.
- Apêndice A - Contém a metodologia para instalação e configuração do *LinuxMCE*.
- Apêndice B - Apresenta o código completo da parte cliente da *gateway*, já integrada no *LinuxMCE* e também do servidor.
- Apêndice C - Expõe as características e configuração da placa *OpenWrt*.
- Apêndice D - Aborda as características e configuração do *Kit Powerline*.
- Apêndice E - Ilustra em forma de uma lista os acrónimos usados no texto

[2]Qualquer sistema computacional com ecrã, usado para controlar o *LinuxMCE*. Explicado mais à frente.

Capítulo 2

Estado da Arte

2.1 Introdução

A domótica, como já foi referido, é um conceito que abarca o controlo e monitorização inteligente de uma habitação, através de métodos automatizados e manuais. Para isso, qualquer sistema destes precisa de possuir actuadores e sensores específicos para cada zona de acção. Por exemplo, ao nível da iluminação, podem existir interruptores (actuadores) e sensores de detecção de luminosidade e, quanto à segurança, detectores de movimento, gás, fumo, inundações e câmaras, entre outros.

A gestão destes dispositivos, ao ser feita de modo individual, apenas com a instalação do estritamente necessário ao seu funcionamento limita, não só o propósito de integrar toda a casa sob um único sistema, como também a automatização do projecto global. Por esta razão, cada vez mais começam a surgir plataformas de gestão de domótica, que suportam um grande conjunto de tecnologias e protocolos, permitem uma integração global de todos os dispositivos na habitação (tanto sensores como actuadores) e fornecem ao utilizador um meio simples para visualizar toda a informação neles contida, bem como os meios para actuar sobre eles. Para além disso, conferem um maior grau de liberdade e personalização ao dono do sistema, pois é possível também criar cenários com vários tipos de acções, conforme variáveis externas (como a luminosidade, o tempo, as horas), interligando sensores e actuadores pertencentes a diferentes zonas de interacção.

O meio de ligação entre os dispositivos na casa e as plataformas de gestão são os variados protocolos existentes na área em estudo, cablados e sem fios que, por vezes, embora não propriamente destinados à domótica, podem ser adaptados e integrados em conjunto com os que o são, para permitir uma maior uniformização e integração das tecnologias já existentes na habitação (por exemplo a *Ethernet* 2.2.1, o *Bluetooth* 2.2.2, o *Wi-Fi - Wireless Fidelity*).

De seguida irá ser feita uma introdução aos protocolos usados na domótica em geral e um resumo de algumas plataformas de gestão existentes, incluindo a usada nesta dissertação, o *LinuxMCE* e, por fim, uma apresentação do *OpenWrt*, que é uma distribuição *open source* para sistemas embutidos.

2.2 Protocolos

Na tabela representada pelas figuras 2.7, 2.8 e 2.9, pode ser consultado um resumo das características principais de todos os protocolos a seguir enunciados e ainda outros, relegando para o texto uma apresentação global daqueles mais importantes, na área em estudo.

2.2.1 Ethernet

A *Ethernet* é o principal protocolo usado actualmente nas redes locais de computadores (LAN - *Local Area Network*). Começou por ser usada apenas em escritórios e implementada em cabos coaxiais com taxas de transmissão na ordem dos 3 Mbps.

A camada física e de dados da comunicação assenta sobre o conhecido modelo OSI (*Open Systems Interconnection*). A parte física está controlada pelo circuito designado *Phy* que tem a função de codificar e descodificar a informação transmitida, enquanto que, a nível de dados, é utilizada uma subcamada do modelo OSI, a MAC (*Medium Access Control*) para controlo de acesso ao meio [2]. Uma comparação da *Ethernet* com o modelo OSI está representada na figura 2.1.

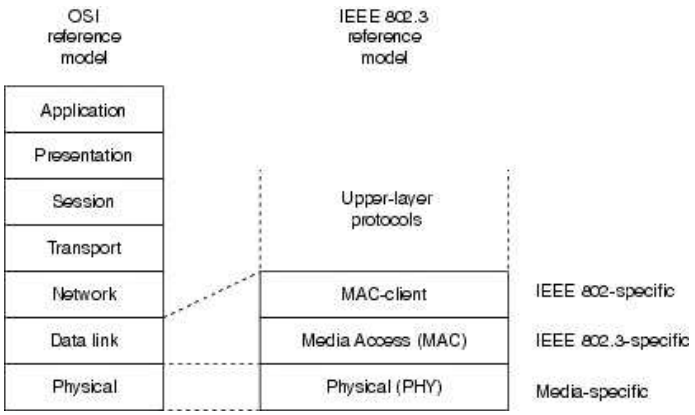

Figura 2.1: Comparação da *Ethernet* com o modelo OSI [2]

Existe ainda um conjunto enorme de *standards* do IEEE, IEEE 802.3 [3] relativos a este protocolo que contêm as características das várias normas actualizadas ao longo dos tempos, bem como usos específicos para a *Ethernet* (por exemplo, *Power Over Ethernet* [4]).

Hoje em dia, estão disponíveis versões optimizadas do protocolo que permitem taxas de transmissão de 100 Mbps para a *fast Ethernet* e até 10Gbps para a *giga Ethernet*. Devido ao uso extensivo do protocolo, este está sempre em actualização e preparado para o futuro, sendo que os padrões de 40Gbps e 100Gbps já estão perto de serem finalizados [5][6].

2.2.2 Bluetooth

O *Bluetooth* é um protocolo aberto sem fios, de comunicação a curtas distâncias. A sua principal linha de lançamento seguiu o objectivo de substituir os cabos normalmente usados para interligar dispositivos fixos e portáteis [7]. As ligações efectuadas entre dispositivos baseiam-se na criação de redes *ad hoc*, mais conhecidas, neste caso, por *piconets* [8]. Cada equipamento presente numa rede destas pode comunicar simultaneamente com outros 7 e pode pertencer a mais que uma *piconet*. Estas são estabelecidas automatica e dinamicamente, sempre que os dispositivos tenham este protocolo activo e se encontrem dentro do raio de alcance da rede. A topologia está ilustrada na figura 2.2.

Actualmente conta com a especificação de 3 normas principais, "*Versão 2.1 Enhanced Data Rate (EDR)*"[9] , "*Versão 3.0 High Speed (HS)*"[10] e "*Versão 4.0 Low Energy*"[11], esta última finalizada recentemente (Abril 2010 [12]) e ainda não disponível no mercado. As duas primeiras são evoluções naturais da versão 1.0 , aumentado o alcance, a largura de banda, a segurança e reduzindo o consumo energético. Já a quarta especificação pretende vir colmatar uma grande falha deste protocolo, ou seja, o seu alto consumo para as aplicações e ambições a que inicialmente se propunha. Assim passará a estar disponível com esta norma, para além do *Bluetooth standard* e o de alta velocidade, uma versão de baixo consumo energético e, consequentemente, reduzida largura de banda.

Como particularidade em relação a outros protocolos, o *Bluetooth* possui, para além dum canal de dados, um canal de voz que pode ser usado em conjunto com o primeiro e é, certamente, por esta característica que ele é mais conhecido actualmente. O uso mais difundido deste protocolo está presente, então, em comunicações audio sem fios entre terminais móveis, computadores e outros equipamentos, e também na transferência de ficheiros de reduzidas dimensões.

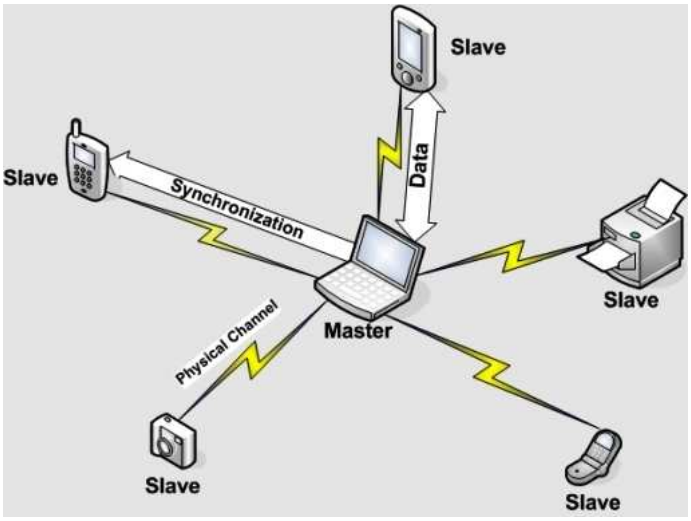

Figura 2.2: Topologia do *Bluetooth* [13]

2.2.3 ZigBee

Uma tecnologia emergente, o *ZigBee* está cada vez a ganhar mais força no campo das WPANs (Wireless Personal Area Networks), muito devido ao facto de se basear no *standard* IEEE 802.15.4-2003.

Apesar de se orientar por um *standard* global aberto, o *ZigBee* é um protocolo proprietário gerido pela *ZigBee Alliance* [14] . A sua utilização não está limitada somente à área da domótica, mas também a aplicações para edifícios comerciais e industriais, energia, telecomunicações, e área da saúde.

Actua em diversas bandas de frequência, conforme a região, e possui uma topologia de rede *mesh*. Com esta disposição da rede são necessários, neste caso, 3 tipos de dispositivos: o coordenador, ZC (*Zigbee Coordinator*), o *router*, ZR (*Zigbee Router*) e o dispositivo final, ZED (*ZigBee End Device*). O primeiro é a estrutura principal da rede e pode até servir de ponto de ligação para outras redes vizinhas; o segundo permite reencaminhar as mensagens entre dispositivos e o último é o dispositivo em si, que dispõe das funcionalidades estritamente necessárias para comunicar apenas com o ZC ou o ZR. Esta simplicidade do ZED permite que o mesmo esteja em modo "adormecido"grande parte do tempo, aumentando em grande escala a bateria útil do sensor [15]. As várias camadas da arquitectura deste protocolo estão ilustradas na figura 2.3.

Parte do sucesso actual prende-se também com uma aposta no reduzido consumo energético, no baixo custo e numa maior simplicidade do protocolo comparativamente, por exemplo, ao *Z-Wave* e ao *Bluetooth*.

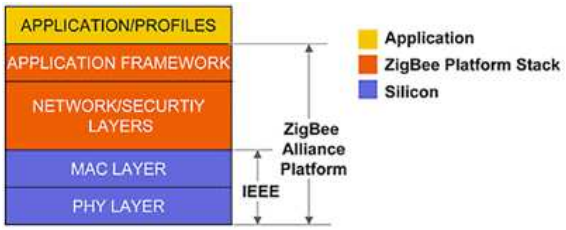

Figura 2.3: Arquitectura do *ZigBee* a nível de rede [16]

2.2.4 G.hn

De entre todos, este é o mais recente e, embora ainda em fase de desenvolvimento pela reconhecida ITU (International Telecommunication Union) [17], destaca-se pelas altas taxas de transmissão previstas e pela ambição de se tornar o protocolo de próxima geração para redes domésticas, ao definir, por fim, um *standard* universal para este tipo de comunicações. Conta já com um grande suporte por parte da indústria, desde a domótica até fornecedores de serviços e vendedores finais de equipamento [18][19][20][21][22].

O protocolo assenta na utilização de um sistema com fios que suporta vários tipos de cabos actualmente usados para comunicações, tais como, telefone, coaxiais e rede eléctrica. A

camada física será implementada através de modulação FFT OFDM (*Fast Fourier Transform Orthogonal Frequency-Division Multiplexing*), com possibilidade de até 4096 QAM (*Quadrature Amplitude Modulation*) e terá um sistema de correcção de erros e imunidade ao ruído avançados. A camada MAC será implementada com base num sistema TDMA (*Time Division Multiple Access*) [23][24].

Devido ao alto débito esperado (na ordem do Gbit/s [25]), segurança e robustez geral do protocolo, a coexistência de aplicações na área da domótica com as necessidades domésticas atendidas actualmente pela *Ethernet*, é uma realidade que pode vir a definir um novo paradigma de integração nas habitações do futuro.

2.2.5 X10

O X10 foi o primeiro protocolo de domótica a ser desenvolvido e é um dos mais comuns nas instalações domóticas.

Decorrente desta situação, está claro que é aquele que menos capacidades tem, incluindo baixa largura da banda e utilização de dispositivos relativamente simples. Não obstante, o seu baixo custo permite a entrada no mundo da automação residencial de forma mais fácil e expedita.

Mais uma vez, a simplicidade do protocolo é notada na trama dos pacotes transmitidos, já que é constituída por um código da casa (limitado às letras A a P), o código da unidade em questão (limitada aos números 1 a 16) e, finalmente, o código do comando a enviar. Estas instruções que podem ser enviadas são pré-definidas pelo X10, como se pode observar na tabela representada pela figura 2.4, e, portanto, limitadas. O número máximo de dispositivos é de 256 devido às restrições impostas para a selecção dos dois primeiros códigos [26].

Utiliza a rede eléctrica para comunicar com os dispositivos, apesar de estar definido também um protocolo rádio sem fios, que necessita de um equipamento para transformar os sinais RF (*Radio Frequency*), de novo para a rede eléctrica.

2.2.6 Insteon

Criado pela empresa *SmartLabs, Inc.* [28], este protocolo surgiu como forma de colmatar as limitações presentes no X10, já descritas na secção 2.2.5, não descurando, no entanto, a retrocompatibilidade com este último [29].

Algumas dessas falhas em conjunto com a exagerada complexidade dos protocolos utilizados na domótica em geral, não permitiam, para a empresa, que o mercado se desenvolvesse e fosse mais além. Assim, ao criar o *Insteon*, basearam-se num sistema *peer-to-peer*, como se pode observar na figura 2.5, que, ao invés das restantes abordagens, não necessita de um *Master*, agindo todos os dispositivos como *peers* da rede, podendo assim receber, transmitir e repetir as mensagens por eles próprios.

Outras melhorias em relação ao X10 são ainda o facto de os dispositivos actuarem como repetidores e assim repetirem todas as mensagens que lhes chegam de forma sincronizada

Lista de Comandos X10		
Código	***Função***	***Descrição***
0000	*All Units OFF*	Desliga todos os equipamentos com o código da casa indicado na mensagem
0001	*All Lights ON*	Liga todos os equipamentos de iluminação (com capacidade de controlo de brilho)
0010	*ON*	Liga um equipamento
0011	*OFF*	Desliga um equipamento
0100	*Dim*	Reduz a intensidade luminosa
0101	*Bright*	Aumenta a intensidade luminosa
0111	*Extended Code*	Código de extensão
1000	*Hail Request*	Solicita resposta do equipamento com o código da casa indicado na mensagem
1001	*Hail Acknowledge*	Resposta ao comando anterior
100X	*Pré-Set Dim*	Permite seleccionar dois niveis pré-definidos de intensidade luminosa
1100	*Extended Data*	Dados adicionais (seguem-se 8 bytes)
1101	*Status is On*	Resposta ao pedido Status Request, indicando que o equipamento está ligado
1110	*Status is Off*	Resposta indicando que o equipamento esta desligado
1111	*Status Request*	Pedido solicitando o estado de um aparelho

Figura 2.4: Conjunto de comandos do X10 [27]

(através da frequência da rede), o que em conjunto com um *acknowledgment* incorporado em todas as mensagens, contribui para a maior robustez do protocolo.

O *Insteon* actua sob a rede eléctrica e/ou através de RF e, tendo em conta a sua simplicidade, a nível de rede e as capacidades de detecção e repetição de mensagens erróneas, tem certamente um futuro risonho.

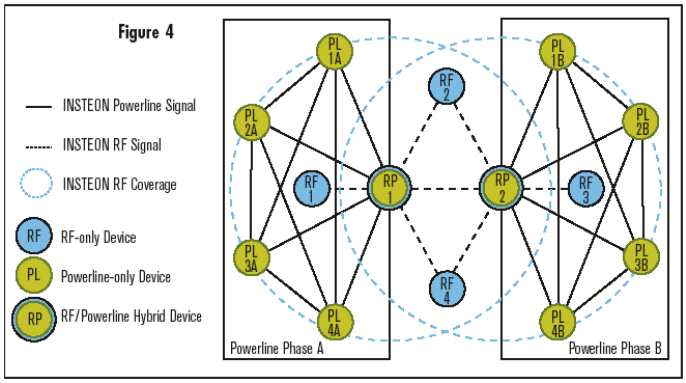

Figura 2.5: Topologia do Insteon [30]

2.2.7 Z-Wave

O *Z-Wave* é um protocolo de comunicações sem fios em RF para aplicações em automação residencial. É gerido por um consórcio, *Z-Wave Alliance* [31] e assenta num modelo fechado/proprietário.

Actua numa banda de frequências, diferente da maior parte dos protocolos de comunicações de rede sem fios, ou seja, abaixo do *Gigahertz*, e, por isso, está mais imune ao ruído, permitindo, assim, uma aplicação global do mesmo em conjunto com outros sistemas sem fios na casa.

A nível de rede, a topologia é fundamentalmente uma rede *mesh* [32], em que os dispositivos ligados actuam como nós da rede e permitem reencaminhar as mensagens por vários caminhos não definidos inicialmente pelo emissor. Para adicionar equipamentos à rede, em primeiro lugar, é necessário registá-los uma vez, executando uma sequência pré-definida nos botões do aparelho.

Esta tecnologia inclui ainda um modo de poupança de energia, no qual o consumo energético é reduzido, o que permite que dispositivos, tais como sensores, tenham uma duração de bateria superior.

2.2.8 LonWorks

Este protocolo é um dos mais abrangentes e não está só presente na área de automação residencial, mas também a nível industrial, de transportes e mesmo em municípios. Tem no historial várias certificações a nível de *standard* de controlo de automação e, recentemente, (Janeiro de 2009) a nível global [33].

À semelhança do Insteon (descrito na secção 2.2.6), é aqui também adoptada uma estrutura de rede *peer-to-peer*, presente na figura 2.6, de modo a descentralizar o sistema e a não ser necessário ter na rede um dispositivo *master*, comunicando assim os dispositivos directamente uns com os outros.

Como meios de transmissão, utiliza a rede eléctrica, cablagem de par cruzado, cabo coaxial, fibra óptica, RF, infravermelhos e *Ethernet*.

Ao contrário de outras soluções no mercado, que apenas disponibilizam partes individuais do sistema a implementar (como camadas físicas protocolares), a *Echelon* [34] apresenta uma plataforma completa de desenvolvimento. Nela estão incluídos o protocolo de comunicação, um processador dedicado [35], transdutores, uma base de dados, ferramentas de acesso aos dispositivos, via serviços *Web* e integração de equipamentos de outros fabricantes.

2.2.9 KNX

O KNX é um dos mais conhecidos e utilizados protocolos na área da domótica. Está actualmente referenciado com *standards* a nível europeu e internacional e é implementado mundialmente por uma quantidade significativa de fabricantes.

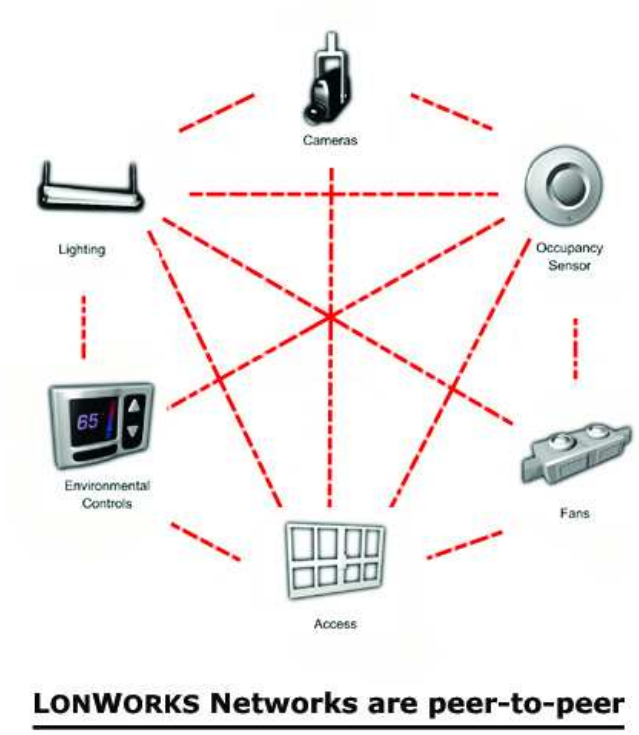

Figura 2.6: Topologia do LonWorks [36]

Ele resulta da junção de três protocolos, o EHS, o BatiBUS e o EIB. Com esta combinação, herdou a pilha protocolar de comunicação do EIB, e parte da camada física, modos de configuração e aplicações finais dos outros dois.

O KNX actua em praticamente todos os meios usuais neste tipo de sistemas, tais como cablagem em par cruzado (a mais usada), rede eléctrica, RF, fibra óptica e *Ethernet*. Tem outra grande vantagem, já que permite ser implementado em praticamente qualquer plataforma computacional, ou seja, é independente do sistema [37].

Como revés, praticamente toda a documentação técnica e ferramentas de desenvolvimento estão disponíveis apenas mediante registo na *KNX Association* [38] e pagamento de uma quota anual.

2.3 Outros Protocolos

Além dos acima mencionados, outros protocolos tais como o Wi-Fi, USB, *FireWire*, CAN, estão presentes na tabela representada pelas figuras 2.7, 2.8 e 2.9, servindo apenas de complemento e termo de comparação.

Protocolo	Taxa de transmissão	Meio de transmissão	Características	Máxima distância	Aplicações
Ethernet IEEE 802.3	10Mbps – 10 Base-T Ethernet	Dois pares entrelaçados com conector RJ-45 (cabo UTP)	Tamanho mínimo da trama de 64 bytes Trama com cabeçalho MAC	100 m	Interconexão para redes locais (LAN)
	100 Mbps Fast Ethernet		Tamanho mínimo da trama de 64 bytes Trama com cabeçalho MAC		
	1000 Mbps Gigabit ethernet	Quatro pares entrelaçados	Tamanho mínimo da trama de 520 bytes Trama com cabeçalho MAC	100 m	
		Fibra óptica simples ou multimodo		316 m	
	10 Gigabit Ethernet	Fibra óptica	Norma IEEE 802.3ae	2 – 15 km	Ligação Ethernet para redes metropolitanas (MANs e WANs)
Bluetooth	V1.2 721 Kbps	Rádio frequência GFSK 2.4-2.4835 GHz	Baixo custo Consumo de energia considerável Redes (*piconets*) *master* – *slave* até 8 dispositivos	Classe 1 100 mW 100 m	Permite a conectividade entre equipamentos (telefones, PCs, impressoras, ratos, teclados) e entre dispositivos de som
	V2.0 2.1 Mbps			Classe2 2.5 mW 10 m	
	V3.0 24 Mbps			Classe 3 1 1mW 1 m	
ZigBee	20 Kbps – 250 Kbps	Rádio frequência BPSK UE (868MHz) US (915MHz) QPSK Worldwide 2.4GHz	Baixo custo Baixo consumo de energia *Wireless sensors networks* WPANs	10-75 m	Domótica Serviços de telecomunicações Construções comerciais e industriais Saúde

Figura 2.7: Primeira parte da tabela de comparação dos protocolos

Wi-Fi **IEEE 802.11**	11-248 Mbps	Rádio frequência OFDM 2.4 GHz IEEE 802.11g/b 5GHz IEEE 802.11a IEEE 802.11n	*Wireless Fidelity* *Access points* (*hotspots*) *Peer-to-peer* *Standard* global (*Wi-Fi Certified*) Encriptação WPE, WPA, WPA2 Consumo de energia considerável	30-100 m	WLANs *Wireless Voice Applications* (VoWLAN ou WVOIP)
USB	V1.0 12 Mbps Half-duplex V2.0 240 Mbps Half-duplex V3.0 4.8 Gbps full-duplex	Par entrelaçado	Topologia em árvore Máximo de 127 dispositivos ligados ao host (*hubs*) *Plug and Play* Comunicação entre *host controller* e *endpoints* através de *pipes* Possibilidade de alimentação externa	5 m Máximo *round trip delay* de 1500 ns	Comunicação entre PCs e periféricos *Mobile devices* (mini/micro usb) Carregar dispositivos (máximo de 500mA, 2.5 watts) Possibilidade de usar em ligações sem fios com o *Wireless* USB
FireWire **IEEE 1394**	400 Mbps – 3.2 Gbps	Par entrelaçado Fibra óptica Wireless Cabo coaxial	Comunica directamente via DMA *Plug and Play* Topologia em árvore *Peer-to-peer* Até 100 dispositivos	4.5-70 m	Interface série para PC e aparelhos digitais de áudio e vídeo Serviços de tempo real Industria automóvel e aeronáutica Redes *ad-hoc* de PCs Carregar dispositivos (até 45 watts)
CAN	125Kbps – 1Mbps	Par entrelaçado	Possibilita a comunicação entre microcontroladores de dispositivos Baixo custo *Multi-master serial bus*	40 – 500 m	Industria automóvel (*Engine Control Unit* (ECU), transmissão, airbags, *cruise control*, ABS, *climate control*)
X-10	50 – 60 bps	Rede eléctrica	Baixo preço Facilidade de uso e instalação Máximo de 256 aparelhos endereçados, e 16 comandos Injecta sinais de 120KHz na rede, e tem um meio de distribuição ruidoso Fraca robustez (simples)	80 m	Domótica

Figura 2.8: Segunda parte da tabela de comparação dos protocolos

Insteon	2.8 Kbps – 38.4 Kbps	Rede eléctrica Rádio frequência	*Peer-to-peer* Não é necessário um *master* na rede Dispositivos actuam também como repetidores	46-80 m	Domótica
Z-Wave	9.6 Kbps – 40 Kbps	Rádio frequência GFSK UE (868MHz) US (908MHz)	Baixo consumo Topologia *mesh* com um máximo de 232 unidades na rede Não transmite imagem/som	30 m	Domótica Eficiência energética Sistemas de segurança *Home Entertainment*
LonWorks	1.7Kbps – 1.28Mbps	Rede eléctrica Par entrelaçado Rádio Frequência Cabo coaxial Fibra óptica	Compatibilidade entre tecnologias LonWorks *Global standard ISO/IEC 14908* *Peer-to-peer* Robustez	1500 – 2700 m	Domótica Automação industrial Transportes Municípios
KNX	1200 -9600 bps	Par entrelaçado (KNX TP) *Power Line* (KNX PL) Rádio Frequência (KNX RF) FSK 868.3MHz IP/Ethernet (KNX IP)	Robustez Flexibilidade Elevado custo Resultou da convergência dos protocolos EIB, EHS e o BatiBUS	300 - 1000 m	Domótica

Figura 2.9: Terceira parte da tabela de comparação dos protocolos

2.4 Plataformas de Gestão Domótica

Um sistema de domótica para automatizar uma casa tem por norma um custo muito elevado e o seu preço de instalação e configuração total varia conforme o(s) tipo(s) de protocolo(s) usado(s) na solução e as secções a controlar e monitorizar seleccionadas.

Em conjunto com este investimento, é necessário também adquirir uma plataforma de gestão de todo o sistema. A maior parte são comerciais e, como tal, apresentam várias desvantagens, como o facto de, muitas delas precisarem de técnicos especializados que cobram valores elevados por cada intervenção, apenas para adicionar um novo componente ao sistema. A falta de modularidade, em grande parte das soluções e a instalação do sistema ter de ser feita em muitos casos, aquando da construção da habitação, completam o leque de

desvantagens deste tipo de abordagem. Por vezes, muitos dos sistemas dedicam-se apenas à área de controlo da habitação, descurando a parte multimédia ou qualquer integração entre as várias zonas de acção sobre as quais se pode actuar.

É óbvio que em relação a uma solução *open source*, as comerciais ganham em alguns campos: o apoio especializado para resolver qualquer problema e a garantia de qualidade da instalação, embora possam apresentar custo elevado, como já foi referido.

Existem várias empresas a comercializarem este tipo de produto, como a *Creston* [39], a AMX [40] e a *HomeSeer* [41]. Em todas elas, o cliente compra um pacote definido previamente, que inclui, não só a plataforma de gestão, como também os dispositivos requeridos para cada área de intervenção apesar de, na última, se poder comprar apenas o *software*. Os dispositivos podem ir desde servidores de áudio e vídeo a interruptores, placas de entrada/saída para sensores e actuadores de iluminação, climatização e segurança, e até ecrãs *touch* e outros equipamentos de interacção com o sistema.

Existe, no entanto, outra possibilidade relativamente a este tópico, que consiste em apostar numa plataforma de gestão gratuita, baseada, na maior parte dos casos, em distribuições de *Linux* [42]. Uma excepção à regra é o *Windows Media Center* [43] que apenas permite gerir aspectos relacionados com multimédia. Em termos de soluções completas, as mais conhecidas são o *LinuxMCE* e o *MisterHouse* [44]. A primeira é muito mais versátil, possui um maior suporte a nível de tecnologias e protocolos, bem como um conjunto de funcionalidades mais elevado e uma configuração dos dispositivos mais facilitada e *user friendly*, em parte devido à maior comunidade em volta dele. O *MisterHouse*, por outro lado, é limitado, tanto a nível de cenários que são possíveis definir, como de equipamentos suportados, sendo o seu *software* mais rudimentar e com menos possibilidades de expansão. Outras opções como o *OpenRemote* [45] encontram-se ainda em estado inicial de desenvolvimento, com limitações ou custos associados a adicionar ao *software* livre.

Para além destas duas soluções completas, existem muitas outras gratuitas, mas que são específicas para um tipo de tecnologia e protocolo, não alcançando assim o objectivo global de gestão desejado neste trabalho.

Ponderando todos estes factores, o *LinuxMCE* é uma solução a considerar, e a escolha certa para este trabalho, pois permite adquirir gratuitamente o programa de gestão de domótica, é mais completo em comparação com os seus concorrentes mais directos, nomeadamente a nível de funcionalidades e protocolos suportados. Na secção seguinte, é efectuada a apresentação desta plataforma gratuita, incluindo as suas características e arquitectura específica.

2.4.1 LinuxMCE

O *LinuxMCE* [46] é uma distribuição *open source* especificamente desenvolvida para se tornar na solução gratuita número um na área da domótica. Até mesmo o seu logotipo se apresenta no sentido de uma solução gratuita nesta área (figura 2.10).

O sistema tem como espinha dorsal a distribuição de *Linux*, *Kubuntu*, em conjunto com uma solução comercial *all-in-one*, *PlutoHome* [47], de automação residencial, entretenimento, segurança e telecomunicações, e tem vindo a evoluir de modo a incorporar cada

vez mais produtos e tecnologias neste domínio. Exemplos disso, e sempre tendo em consideração a orientação livre do *software*, foram incorporados outros projectos *open source* de grande relevância nas áreas em que actuam, como o *Xine* [48] (leitor de multimédia), *Asterisk* [49] (para gestão de comunicações de voz), *MythTV* [50] (um *frontend* para captura de vídeo), entre outros.

Possui uma arquitectura de rede baseada em *Linux* e um sistema simples de troca de mensagens entre dispositivos permitindo, apesar dessa simplicidade, uma integração de alto nível com outros produtos.

Actualmente, encontra-se na versão 8.10b2, com uma versão RC (*Release Candidate*) esperada para breve.

A nível de desenvolvimento, está disponível um repositório SVN [51] (*SubVersioN*), com o código fonte actualizado constantemente, uma *Wiki* [52], fórum e canal de IRC (*Internet Relay Chat*) para obter mais informação do sistema, bem como tirar algumas dúvidas.

Como *open source* que é, o *LinuxMCE* está dependente da boa vontade e dedicação da comunidade, tanto a nível de correcção de erros e implementação de novas funcionalidades, mas também da documentação (algo parca a nível de desenvolvimento) existente.

Figura 2.10: Logótipo do *LinuxMCE* [53]

2.4.1.1 Funcionalidades

As funcionalidades proporcionadas pelo sistema dividem-se em várias áreas:

- Multimédia (Figura 2.11):

- Organizar todo o conteúdo multimédia através de diferentes identificadores

- Ver/ouvir conteúdo multimédia em qualquer divisão
- Possibilidade de qualquer conteúdo multimédia seguir o utilizador pelas várias divisões
- Detecção automática de novos itens multimédia presentes na rede
- Controlo de todos os dispositivos de áudio e vídeo através do *LinuxMCE*
- Cenário conjunto com a parte de iluminação e telecomunicações, para gerir luzes e chamadas conforme acções pré-definidas.

Figura 2.11: Parte de Multimédia do *LinuxMCE* [54]

- Telecomunicações (Figura 2.12):

- Detecção *Plug and Play* de telefones VoIP (*Voice Over Internet Protocol*)
- Controlo da casa através de qualquer telefone
- Redireccionamento das chamadas consoante o modo definido para a casa na parte de segurança
- Apresentação de um ID da pessoa que está a telefonar no ecrã, com possibilidade de interromper a reprodução de conteúdo multimédia
- *Voicemail* interactivo

- Segurança (Figura 2.13):

Figura 2.12: Parte de Telecomunicações do *LinuxMCE* [55]

- Monitorizar câmeras de vigilância
- Activar ou desactivar o alarme, usando os *Orbiters*
- Activar ou desactivar o alarme, usando sensores de proximidade
- Tirar fotos automaticamente, quando os sensores detectam movimento e actuar em conjunto com a parte de iluminação para ligar as luzes
- Em conjunto com a parte de telecomunicações, se for detectada uma brecha na segurança, pode ser feita uma chamada para um número pré-definido e enviar também uma imagem para o telefone
- Em conjunto com a parte de multimédia, falar com o intruso e mesmo obter vídeo em directo do mesmo

- Iluminação (Figura 2.14):

- Ligar e desligar luzes conforme se entra e sai das divisões
- Aceder remotamente a todo o sistema de iluminação através da interface *Web* do *LinuxMCE*
- Reduzir o impacto a nível energético, usando vários cenários de iluminação
- Em conjunto com a parte de telecomunicações, possibilitar que as luzes sejam ligadas e desligadas quando o telefone tocar, de modo a ajudar as pessoas com deficiências auditivas

Figura 2.13: Parte de Segurança do *LinuxMCE* [56]

Figura 2.14: Parte de Iluminação do *LinuxMCE* [57]

- Climatização (Figura 2.15):

- Controlar dispositivos de climatização com base em cenários pré-definidos
- Controlar dispositivos de climatização directamente na imagem da planta da casa
- Fechar e abrir janelas e estores automaticamente, consoante a hora do dia

Figura 2.15: Parte de Climatização do *LinuxMCE* [58]

2.4.1.2 Componentes do sistema

No *LinuxMCE*, o sistema é constituído basicamente por três componentes físicos principais, o *Core PC (Personal Computer)*, os MDs (*Media Directors*) e os *Orbiters*. O *Core PC* e os MDs estão interligados por LAN através de *switches* ou eventualmente por *wireless*[1].

O *Core PC* tem a função de servidor e, portanto, de controlar e correr todas as aplicações necessárias na rede. As definições de todo o sistema e as informações direccionadas aos dispositivos são guardadas e geridas aqui, mesmo não estando a correr localmente, mas noutros computadores da rede. Isto permite que esteja disponível uma função *plug and play* e que, assim, os dispositivos ligados ao sistema, em qualquer parte da casa, sejam automaticamente detectados. Este servidor disponibiliza ainda uma página de administração na *Web* ilustrada pelas figuras 2.16 e 2.17, em que se pode controlar todo o sistema, não só a partir deste computador, mas também de qualquer outro, na rede. Assim, se for desejado, o PC destinado a esta função pode ter apenas a unidade de processamento, não necessitando de teclado, rato, ou mesmo ecrã. Por fim, este *Core PC* contém a possibilidade de correr serviços de *netboot*, o que facilita a integração de outros computadores nesta distribuição, pois assim não precisam de correr um sistema operativo, podendo fazer o *boot* directamente, a partir da rede.

Os MDs são todos os outros computadores presentes no *LinuxMCE* e, além de poderem ter ligados outros dispositivos (embora controlados inequivocamente pelo *Core*), têm como principal função actuar como terminais para visualização de conteúdo multimédia[2] e interacção com toda a casa (isto porque existe um *Orbiter* incorporado em todos os MDs, representado na figura 2.18). Todo o conteúdo a visualizar é gerido pelo servidor, limitando-se assim os MDs a disponibilizar essa informação para os sistemas de áudio e vídeo aos quais estão ligados.

Por fim, os *Orbiters* são os dispositivos mais interessantes para controlar o *LinuxMCE*, a nível de utilizador e, consequentemente, a casa, pois não precisam de ser muito complexos, mas sim portáteis e, por conseguinte, apresentam uma grande mobilidade e conveniência. Permitem obter informação dos diversos sensores na casa, agir sobre os actuadores e mudar os diversos cenários disponíveis em cada divisão da habitação. Para funcionarem, precisam

[1] embora seja recomendado uma ligação com fios devido ao tipo de tráfego (muitas vezes vídeo) a circular na rede

[2] o Core também pode actuar em modo híbrido como o MD, acumulando assim as duas funções

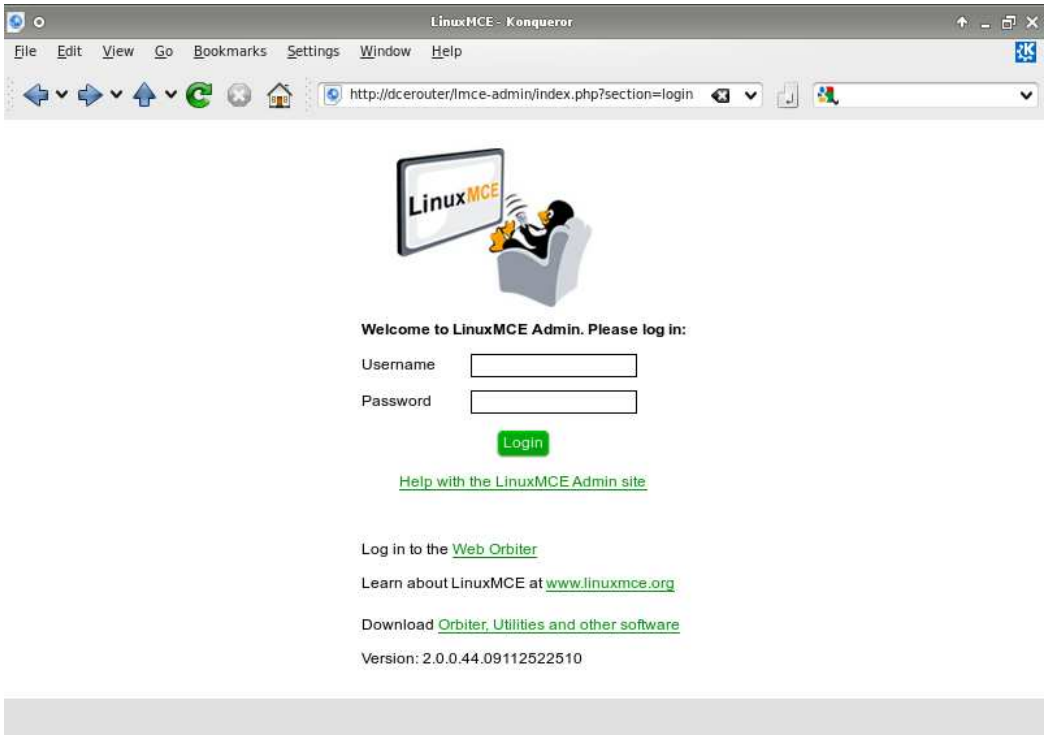

Figura 2.16: Página de login na interface de *WebAdmin* do *LinuxMCE*

de estar ligados ao *LinuxMCE* através dum MD ou, então, conectados à LAN interna, através dum AP *wireless*.

A nível de *hardware*, como é de esperar, o *Core* é a parte do sistema que necessita de maior poder de processamento, daí que o uso de um bom processador e uma quantidade razoável de memória seja o mais indicado. Para além disso, é imperativo ter uma interface *Ethernet Dual NIC (Network Interface Controller)*[3] devido à arquitectura imposta pelo sistema (um NIC destinado exclusivamente à rede interna de automação e outro para a rede externa, como a Internet), e também várias portas de expansão, pois é aqui que se vai ligar uma grande parte dos sensores, placas de TV e aquisição de imagem, entre outros.

Os *Media Directors*, devido à função de *netboot* do *Core PC*, podem ser meros *thin clients*, alojados atrás duma televisão/ecrã ou portáteis/desktops completos com especificações semelhantes à de qualquer PC para *Home Theater*. Têm de possuir ainda as entradas/saídas necessárias à interligação com o sistema de A/V (*Audio/Video*) desejado, e com os métodos de controlo requeridos (como comandos, ratos, dispositivos *bluetooth*, entre outros).

Relativamente aos *Orbiters*, podem ser desde telemóveis, PDAs (*Personal Digital Assistants*), *tablets*, comandos, e até telefones VOIP com ecrã. Outra opção passa por instalar o *software* dum *Orbiter* num PC normal ou aceder através de *Web*, à interface dum *Web*

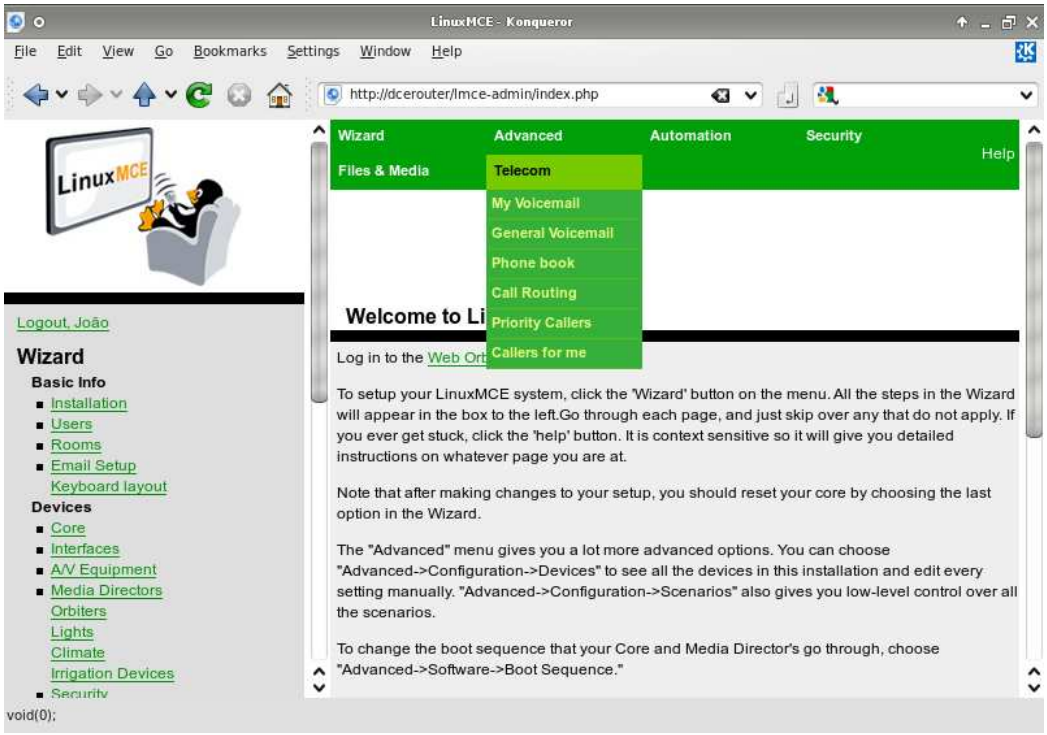

Figura 2.17: Página da interface de *WebAdmin* do *LinuxMCE*

Figura 2.18: Ecrã do *Orbiter* presente nos MDs [53]

Orbiter, ilustrado na figura 2.19, sem instalar qualquer programa.

Destes componentes, o único que é estritamente essencial é o *Core PC*. Os outros podem ou não estar presentes e apenas adicionam funcionalidades ao sistema, como já vimos acima.

Nas figuras 2.20 e 2.21 é possível observar exemplos de algumas configurações tipo para

[3]isto é, com duas portas de rede

Figura 2.19: Ecrã do *WebOrbiter*

o *LinuxMCE*.

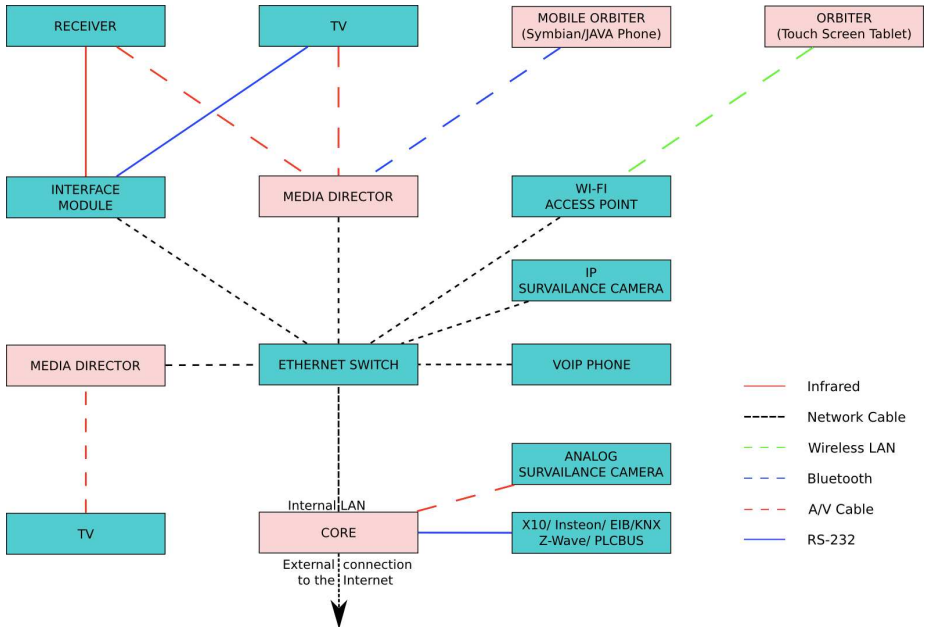

Figura 2.20: Diagrama da configuração do *LinuxMCE* com Core PC dedicado [59]

Uma boa abordagem para a disposição destes componentes na casa (exceptuando os *Orbiters* que são os componentes mais móveis), passa por colocar o *Core PC* "escondido"num

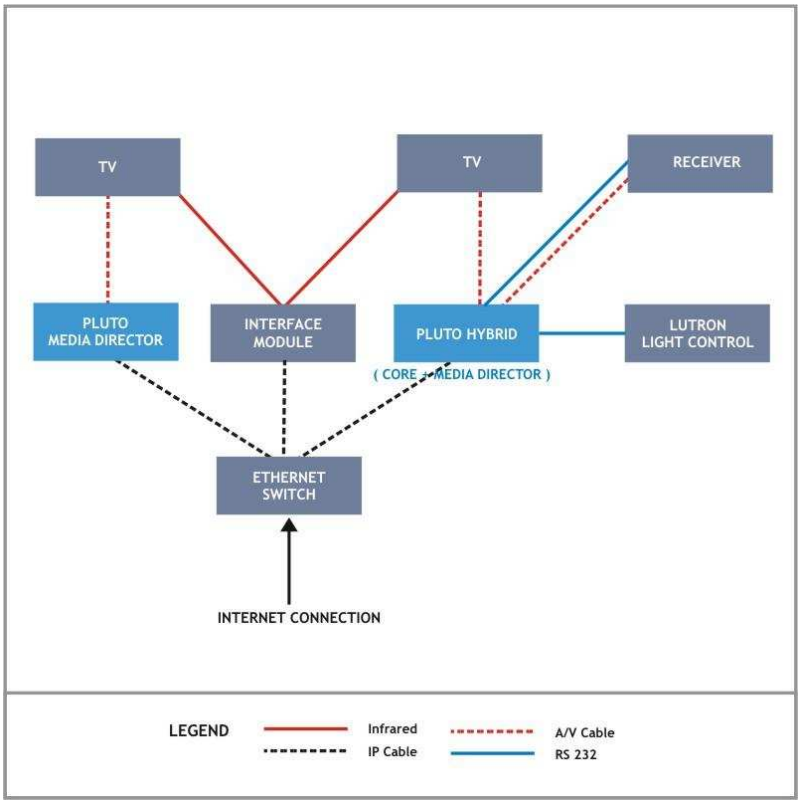

Figura 2.21: Diagrama da configuração do *LinuxMCE* com Core PC híbrido [60]

qualquer lugar da habitação, se em modo dedicado, pois não é necessário, posteriormente à instalação do sistema, o acesso físico ao mesmo. No caso de actuar em modo híbrido, deve colocar-se o mesmo numa divisão, na qual se pretenda disponibilizar conteúdo multimédia (por exemplo na sala). Quanto aos MDs, o usual será colocar um em cada uma das restantes divisões propensas a servirem de centro multimédia (como os quartos, cozinha, escritórios).

2.4.1.3 Arquitectura

A nível de arquitectura, o *LinuxMCE* baseia-se numa abordagem modular, como se pode observar na figura 2.22. A parte essencial de todo o sistema, chama-se *DCERouter* (*Data Commands Events Router*) e não é mais do que um programa, que simula um *router* genérico de envio e recepção de mensagens. Assim, este não contém qualquer tipo de informação relativa ao *LinuxMCE* em si, mas apenas encaminha as mensagens trocadas, entre os dispositivos, que correm em programas independentes dele. Por fim, existe ainda um tipo especial de dispositivos, os *plugins*, que, em vez de correrem em programas separados do *DCERouter*, estão presentes no seu espaço de memória, como bibliotecas dinâmicas.

De resto, todas as definições do sistema, dados de cada utilizador, da instalação habi-

tacional e dos dispositivos estão armazenados numa base de dados SQL (*Structured Query Language*) central, que disponibiliza a informação sempre que for solicitada, tanto pelo utilizador, como pelo *DCERouter* (e consequentemente por algum dispositivo). Qualquer alteração dos dados já referidos é comunicada e sincronizada com a base de dados, através da ferramenta *sqlCVS*, que permite realizar operações atómicas sobre eles e monitorizar quem fez essas mudanças, elevando assim o nível de segurança e robustez do sistema em geral.

De notar que, tanto o *DCERouter* como os programas específicos de cada dispositivo e até a base de dados SQL, estão armazenados e a correr no *Core PC* que, como já foi explicado, é a unidade de processamento de todo o sistema.

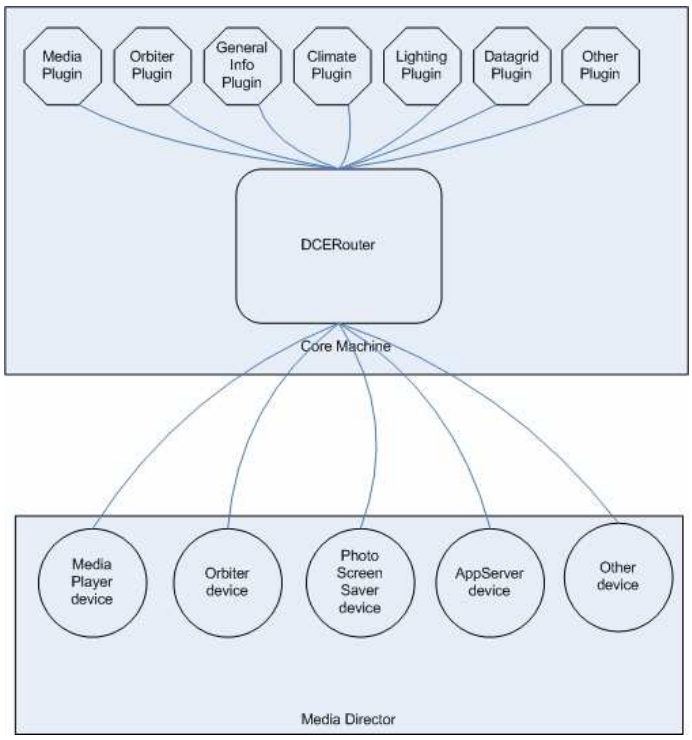

Figura 2.22: Arquitectura do *LinuxMCE* [61]

O *DCERouter*

Numa fase inicial, o papel do *DCERouter* é fornecer os dados de configuração aos dispositivos que se conectam a ele (desde que tenham o sistema de mensagens DCE implementado) e, depois disso, ficar responsável por receber e encaminhar qualquer mensagem de/para o dispositivo. Os tipos mais comuns de mensagens são os comandos e os eventos, incorporados no sistema de mensagems integrado no *LinuxMCE* (DCE), a explicar posteriormente.

Um exemplo deste tipo de comunicações seria uma campainha de porta (actuando como

um dispositivo e ligada à rede) lançar um evento contendo a informação que o botão foi pressionado, evento esse que iria ter ao *DCERouter*, o qual o reencaminharia para um *plugin* destinado a lidar com esta situação. Poderia ainda dar-se o caso do *plugin* querer enviar um comando a um leitor de multimédia, e novamente passaria primeiro pelo *router*, só depois chegando ao seu destino final.

Como se pode ver, os dispositivos DCE nunca comunicam directamente uns com os outros, mas sim através deste elemento fundamental que fomenta toda a simplicidade e modularidade da arquitectura.

O *DCERouter* permite ainda que os dispositivos digam que querem pré-processar mensagens que cumpram certos critérios. Por exemplo, um *plugin* de segurança pode necessitar de obter todos os eventos relacionados com esse campo, para depois actuar sobre outros equipamentos, mesmo que sejam controlados por *plugins* de outra área, como iluminação. Isto denota outra característica dos *plugins*, pois ao serem carregados no espaço de memória do *DCERouter*, partilham também a memória uns com os outros, podendo assim chamar métodos e funções que não apenas os deles.

A nível funcional, o *router*, descrito pela figura 2.23, apenas abre uma porta para as ligações dos vários dispositivos. Depois, quando um dispositivo se liga, envia o seu ID ou *Mac Address*, o *DCERouter* procura esse ID, na base de dados, e devolve-lhe a sua configuração. De seguida, o dispositivo abre um mínimo de duas ligações via *sockets* para o *router*, uma dedicada à recepção de mensagens (normalmente comandos) e outra para as enviar (tipicamente eventos). A biblioteca DCE abre ainda uma terceira ligação de modo a existirem dois *sockets* de saída, um para enviar eventos e o outro para enviar outro tipo de mensagens, como pedidos e comandos para outros dispositivos.

O sistema de mensagens DCE

O DCE é um protocolo leve de rede, que permite que qualquer dispositivo aceda à sua configuração (isto é, aos seus dados específicos), a partir de uma base de dados central; enviar e receber comandos, e accionar e responder a eventos.

Este protocolo está escrito em C++, é baseado em *sockets* e corre actualmente sobre *Linux*.

As mensagens podem ser enviadas para mais do que um dispositivo e contêm informação sobre o dispositivo de destino, o tipo de mensagem (comando, evento), o identificador da mensagem e um número variável de parâmetros opcionais.

Todas elas são enviadas através das funções *SendMessage* ou *QueueMessageToRouter*, quer sejam comandos ou eventos.

2.5 OpenWrt

O *OpenWrt* [63] é uma distribuição gratuita de *Linux* para sistemas embutidos, tais como *gateways* e *routers* residenciais. Começou por ser desenvolvido para apenas um *router*, mas posteriormente foi expandido para várias arquitecturas e *chipsets*, tais como,

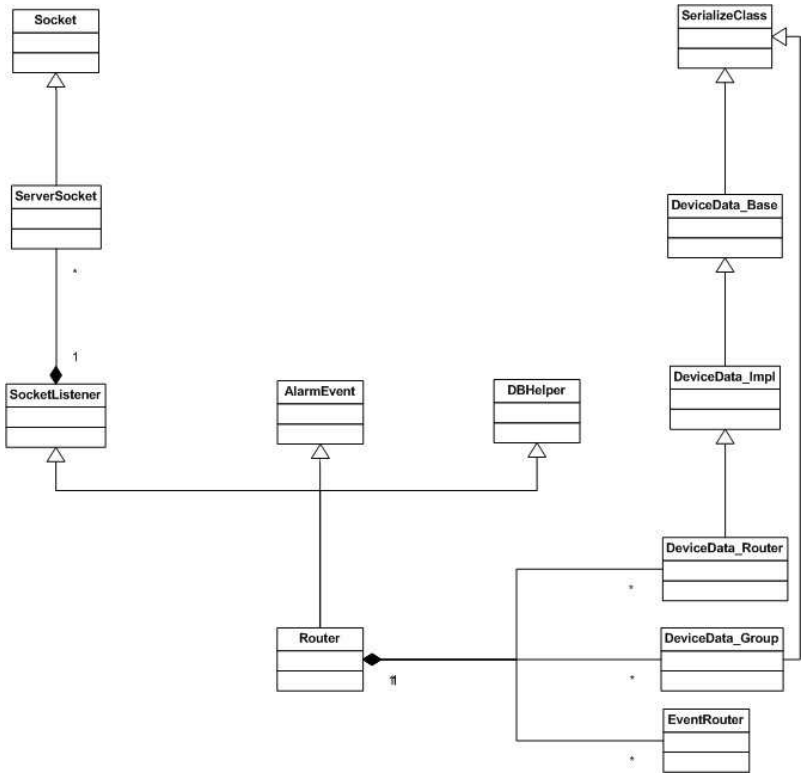

Figura 2.23: Estrutura do *DCERouter* no *LinuxMCE* [62]

atheros, *avr32*, *ixp4xx* e *x86* entre outras.

O objectivo do projecto é fornecer uma estrutura para criar um *firmware* adaptado às necessidades de cada utilizador. Para isso, a distribuição apresenta um sistema de ficheiros totalmente editável com um gestor de pacotes incorporado. A nível de interacção com a distribuição, está disponível a linha de comandos através de SSH (*Secure Shell*) ou *telnet* e uma interface *Web*.

Como funcionalidades principais apresenta todas as normais nos equipamentos de origem, tais como serviços de DHCP (*Dynamic Host Configuration Protocol*), encriptação das ligações sem fios, WEP (*Wireless Encryption Protocol*), WPA (*Wi-Fi Protected Access*) e WPA 2 (*Wi-Fi Protected Access 2*), reencaminhento através de porta, UPnP (*Universal Plug and Play*), configurações avançadas da *firewall*, QoS (*Quality of service*) para vários tipos de tráfego, configuração do dispositivo como repetidor, AP ou *bridge* sem fios, serviços de DNS (*Domain Name Service*) dinâmicos, partilha de ficheiros, ligação de impressoras e outros equipamentos através de portas USB (*Universal Serial Bus*) e monitorização, em tempo real da rede.

A nível de pacotes que se podem instalar, tanto depois da instalação como na fase de compilação do *firmware*, existem bastantes [64], para específicas necessidades. Para além destes oficiais, existem muitos outros, visto este ser um projecto para a comunidade, e, por conseguinte, com muita participação da mesma.

Actualmente na versão 10.03, é considerada a melhor solução de *firmware* dentro da sua área, ultrapassando as outras soluções a nível de extensibilidade, robustez e *design*.

Capítulo 3

Gateway LinuxMCE - OpenWrt

3.1 Introdução

Como ponto de partida, o objectivo do trabalho passava por integrar um ou vários APs 802.11x (possuindo interface *Fast/Gigabit Ethernet*), com sensores diversos em redes já existentes numa habitação, mas que pudessem ser usadas no âmbito da domótica, como se pode observar na figura 3.1.

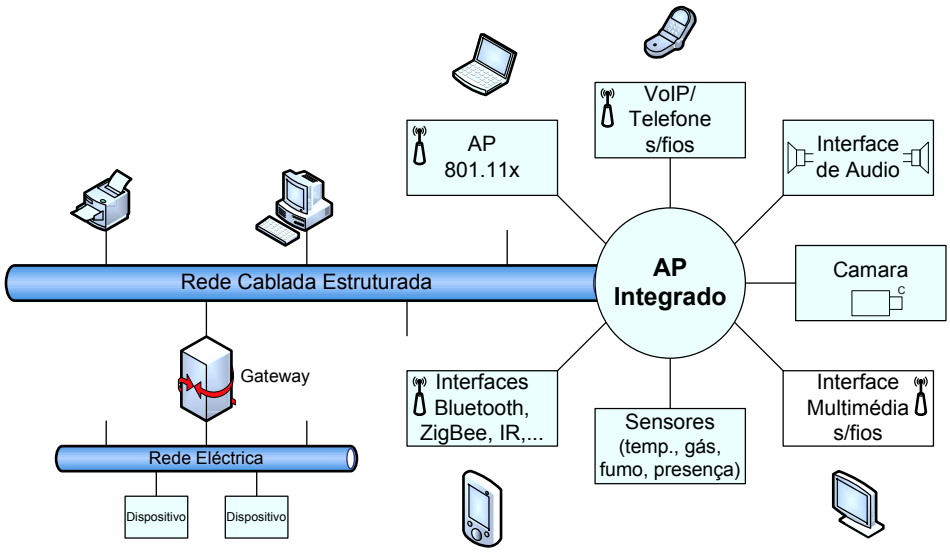

Figura 3.1: Diagrama de uma possível rede domótica

O recurso às redes de comunicação de dados genéricas, já disponíveis numa casa, é uma maneira de simplificar a arquitectura e baixar os custos de instalação de um sistema deste género que, de outra forma, teria de ser baseado em vários tipos de protocolos para cada

área de acção e para cada fabricante. A junção de redes não cabladas e cabladas justifica-se pois, apesar das primeiras poderem ser usadas com enormes vantagens na implementação de sistemas domóticos em edíficios pré-existentes, as redes cabladas são também uma solução interessante quando instaladas durante a construção, e até posteriormente, com recurso por exemplo, à rede eléctrica, possuindo enormes vantagens ao nível da segurança, desempenho e manutenção.

As funções deste "AP Integrado"vão permitir, a montante, a utilização de uma rede de dados cablada e, a jusante, disponibilizar uma rede de dados não cablada (ambas *standard* de uso geral) e integrar os componentes e mecanismos para fornecer os serviços adicionais, assim como as respectivas interfaces para acesso à rede. Para atingir este objectivo, o *LinuxMCE* surge como mediador entre a habitação e os sensores disponíveis e que, em conjunto com a *gateway* implementada, gere todo o sistema e disponibiliza a sua informação aos utilizadores da casa.

Os APs são equipamentos de rede *standard* que devem permitir adicionar dispositivos, métodos de entrada e saída de dados e apresentar uma grande customização via *software*, de modo a facilitar a integração com a plataforma de gestão, que é igualmente flexível. Num cenário ideal, desejar-se-ia que o AP se comportasse como parte integrante do *LinuxMCE*, e os diversos dispositivos deveriam correr totalmente no AP em vez de, no *CorePC*. No entanto, devido à quantidade de trabalho, tempo requerido, complexidade e o facto de o *LinuxMCE* não suportar a sua instalação em plataformas mais modestas, teve de se seguir uma abordagem alternativa, mas igualmente válida para os objectivos pretendidos. Assim, partindo desta plataforma de gestão aberta, integraram-se os sensores dos APs no *LinuxMCE* e criou-se um denominador comum de comunicação entre esta distribuição e os *Access Points* a correr *OpenWrt*, a *gateway*.

A necessidade da execução de uma *gateway* prende-se com o facto de não existir um método simples e eficaz de integrar os dispositivos no *LinuxMCE*, como já foi falado. Assim, foram utilizados *sockets* como meio de comunicação entre o *LinuxMCE* e o *OpenWrt* (nos APs), de modo a atingir uma simbiose adequada entre as duas plataformas, permitindo facilmente o suporte e integração de dispositivos de ambos os lados.

De modo a executar esta parte do trabalho, estabeleceu-se uma arquitectura simples para o sistema, procederam-se aos passos necessários para a criação e instalação de novos dispositivos no *LinuxMCE* e, finalmente, implementaram-se as funções da *gateway*.

3.2 Arquitectura

Devido à complexidade de implementar o sistema de mensagens DCE nos sensores, optou-se por seguir uma aproximação mais simples, baseada num sistema de *sockets* em C/C++.

O conceito geral das ligações da *gateway* é apresentado na figura 3.2. Segundo uma perspectiva *top-down*, no lado esquerdo está presente o sistema *LinuxMCE*, e do lado direito os APs com *OpenWrt*. Quanto ao *LinuxMCE*, são ilustrados os seus componentes característicos: o *CorePC*, o *DCERouter* e paralelamente os *Orbiters*. Os dispositivos

propriamente ditos (sensores neste caso) estão ligados do lado dos APs, os quais também correm o seu código.

A *gateway* é representada pelas setas entre estes dois subsistemas, e a interligação entre eles é feita pela mesma através de um sistema de *sockets*. Ela foi construída sobre uma topologia cliente-servidor. Na parte referente ao *LinuxMCE*, corre a parte do(s) cliente(s), específica a cada dispositivo e no lado dos APs é executada a porção de código do servidor (que gere os sensores em cada AP).

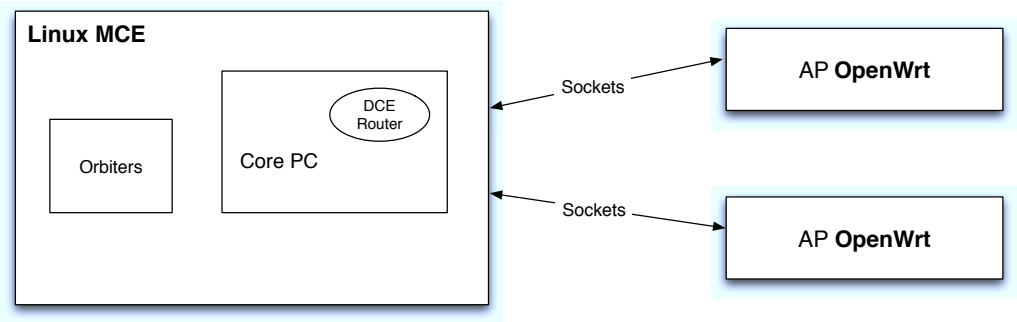

Figura 3.2: Diagrama dos componentes de todo o sistema

A parte do servidor desenvolveu-se de modo a simular a actividade de sensores genéricos, com as respectivas funções. Este corria localmente num portátil e actuava em conjunto com o código do cliente, que é efectivamente a parte residente no *LinuxMCE*. Posteriormente, o código do servidor foi migrado para os APs para integrar o demonstrador final.

Já a nível temporal, como se pode observar na figura 3.3, a arquitectura da *gateway* contém várias etapas bem definidas.

Tudo se inicia com um pedido (accionado pelo utilizador), do lado do *LinuxMCE*, direccionado a um sensor específico. Esta requisição pode variar desde um comando para ligar o dispositivo ou qualquer outra acção para actuar sobre ele, até um simples reconhecimento do estado, da posição ou outra variável relevante para o sensor.

Depois do pedido ser efectuado, o cliente reconhece-o automaticamente (algo que é intrínseco ao *LinuxMCE*) e executa a porção de código referente a esse comando. Nesta fase, é aberta uma ligação para o servidor e enviado o ID específico do sensor em questão. O cliente aguarda pela confirmação da existência do ID (enviada pelo AP) e, em caso afirmativo, envia neste momento o comando que foi requisitado pelo utilizador. Posto isto, o servidor verifica e executa-o, reportando de volta o resultado, como está ilustrado na figura 3.4. Finalmente, o cliente recebe o resultado do comando e fornece-o ao *LinuxMCE*, via um *popup*.

De modo a concluir o processo, o cliente fecha a ligação estabelecida no início, ao contrário do AP, que continua a executar o código, de modo a estar disponível para fornecer informação a qualquer instante.

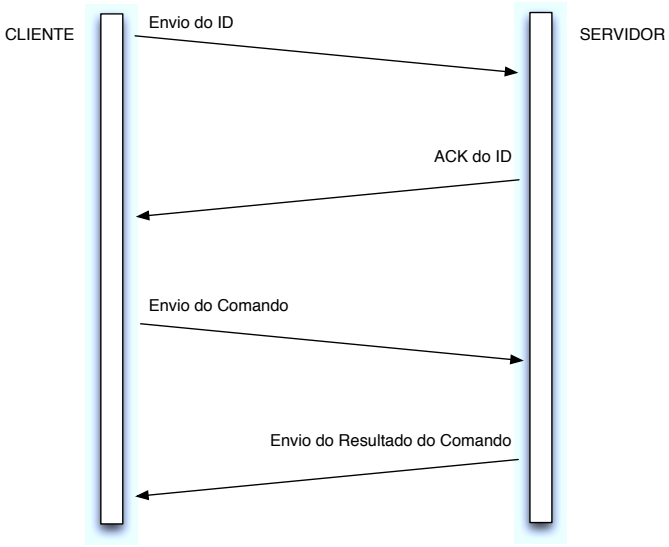

Figura 3.3: Diagrama temporal da *Gateway*

Em resumo, com este tipo de arquitectura, é possível enviar variáveis obtidas a partir dos sensores e actuar sobre eles de maneira simples e inequívoca. O tipo de variáveis a controlar e observar pode ir desde números fixos (ID do sensor), variáveis de estado ("0"ou "1", "ON"ou "OFF") e dinâmicas.

3.3 Criação de *templates* e compilação de novos dispositivos

A primeira etapa consiste em instalar e configurar o *LinuxMCE* com as instruções indicadas no apêndice A.1. Depois de ter o *LinuxMCE* configurado, o passo natural foi perceber como se adicionavam dispositivos ao sistema. A sua arquitectura exige que todo e qualquer equipamento seja acrescentado via um modelo (*template*), previamente criado, que define o dispositivo, ou seja, os seus dados, comandos e eventos.

Para criar um *template*, é necessário aceder à interface de *WebAdmin*, através do endereço `http://dcerouter`, ir ao menu "*Advanced - Configuration - Device Templates*"(figura 3.5). A partir daqui, selecciona-se o fabricante (*Manufacturer*) do dispositivo a adicionar, ou cria-se um novo, escolhe-se a categoria a que este pertence (*Device Category*) e carrega-se em (*Add device template*).

De seguida, é aberta uma janela nova para editar o modelo (figura 3.6) onde, no caso dos sensores envolvidos neste projecto, é preciso definir o nome do sensor, tirar o "visto"da opção "*Implements DCE*", activar a que diz "*Is IP Based*"e seleccionar como método de comunicação, "*Comm Method*"*Ethernet*. No campo "*Device Data*", adicionam-se os parâmetros específicos do sensor, neste caso, são apenas "*Port/Channel*"e a opção

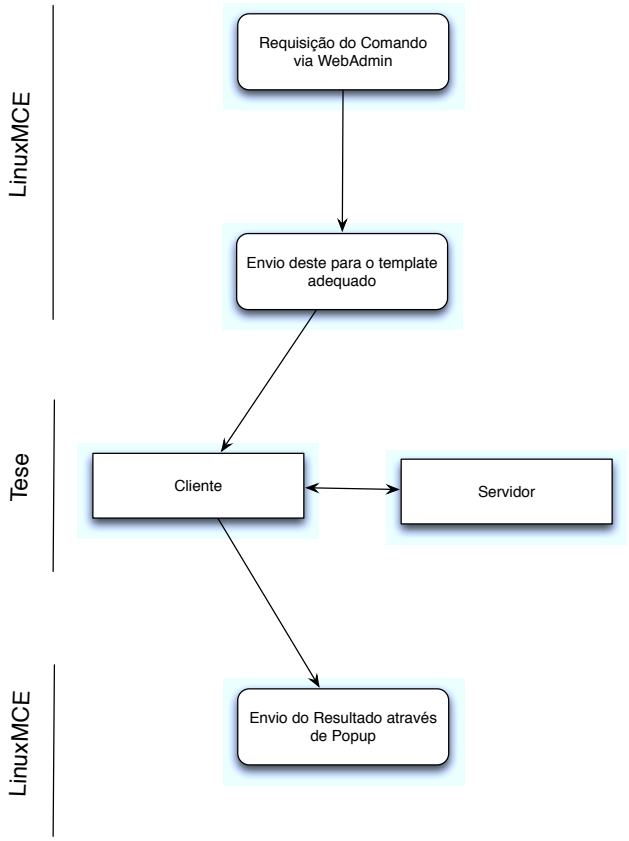

Figura 3.4: Diagrama geral da *Gateway*

"*PK_FloorplanObjectType*"que indicam, respectivamente, a porta onde o sensor está a correr e o tipo de objecto que deve aparecer na planta da casa para o ilustrar.

Das opções que restam, também específicas a cada dispositivo, é necessário indicar os comandos e eventos que cada um deles implementa (figura 3.7). Para finalizar o processo, basta gravar as alterações neste ecrã, aceder através da *WebAdmin* a "*Advanced - sqlCVS - Update*", e executar um *SqlCVS Update* com os parâmetros origem, para actualizar a base de dados carregando em *Next*.

Após esta etapa, resta saber como gerar o esqueleto do código a partir do modelo e como compilar o dispositivo.

Em termos gerais, é preciso instalar um ambiente de desenvolvimento no *Core PC* de modo a poder compilar código para os novos dispositivos.

Posto isto, é preciso utilizar as ferramentas *DCEGen* e *sql2cpp* para gerar o esqueleto do código do dispositivo, a partir do *template* definido anteriormente.

O primeiro programa serve para, a partir do modelo do sensor, criar uma pasta contendo os vários ficheiros necessários para a implementação do mesmo, isto é, os *.cpp* e os *.h*. O

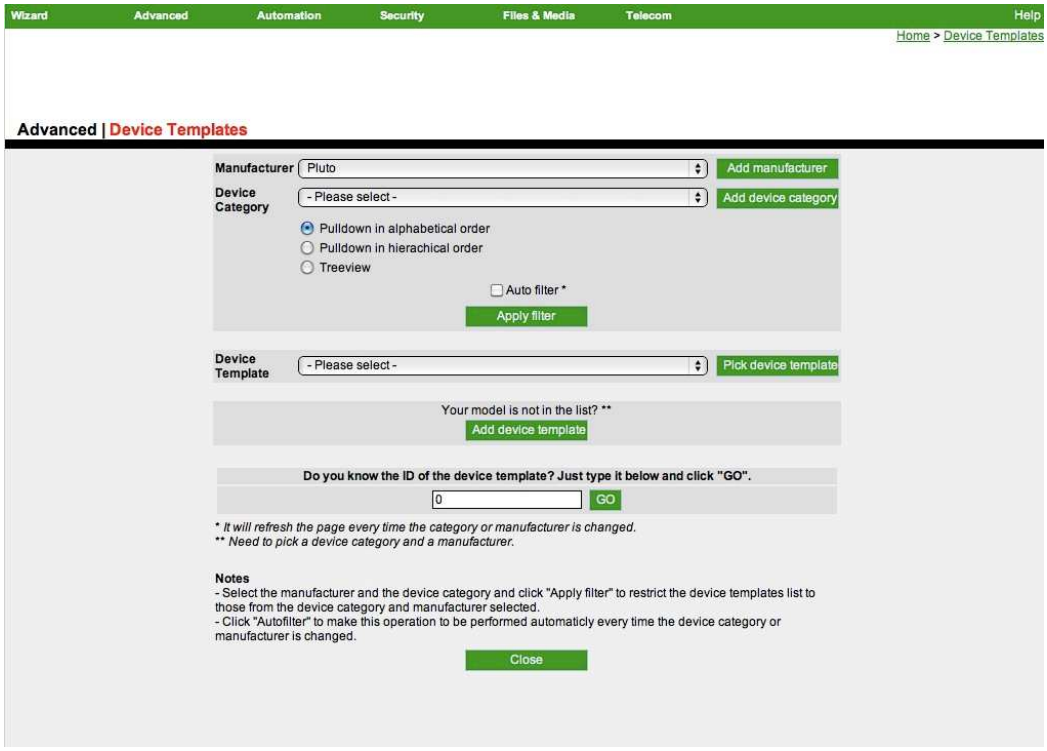

Figura 3.5: Página genérica dos *device templates* do *LinuxMCE*

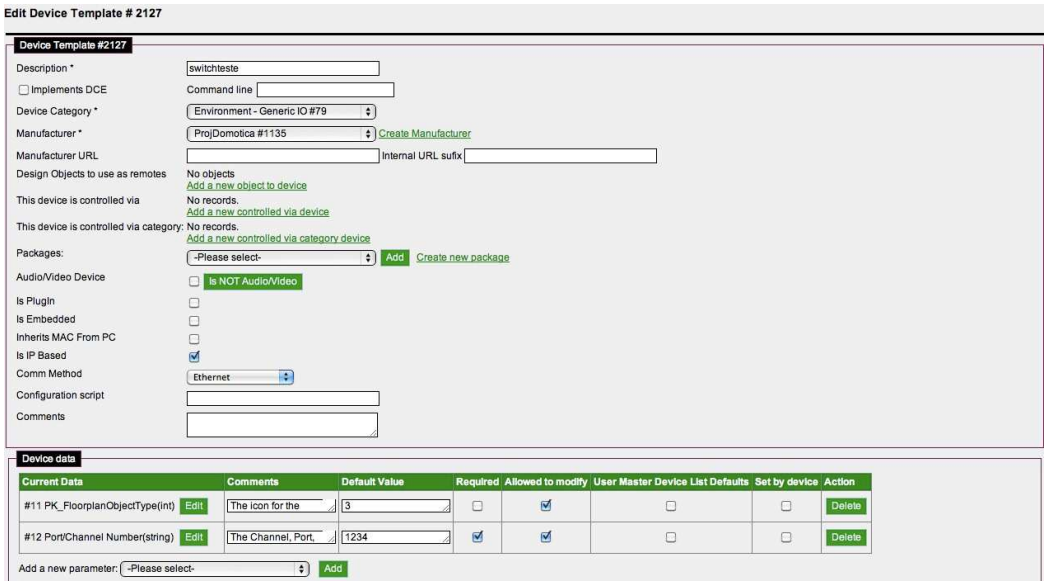

Figura 3.6: Primeira parte da página de um novo *device template* do *LinuxMCE*

nome da pasta é igual ao nome do dispositivo e os ficheiros contêm a declaração de todos os comandos previamente definidos que, originalmente, apenas imprimem uma *string* para a

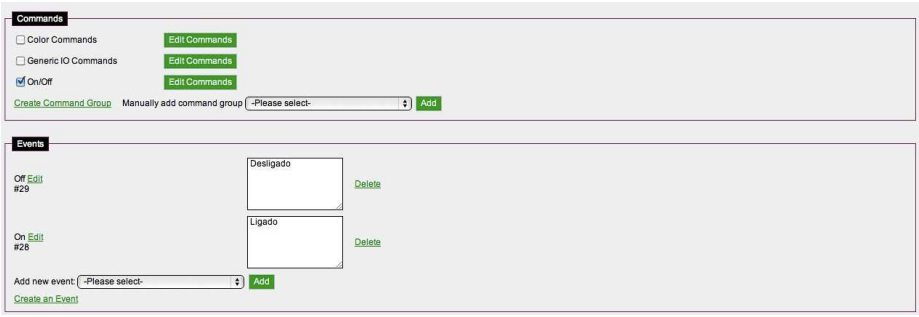

Figura 3.7: Segunda parte da página de um novo *device template* do *LinuxMCE*

janela de terminal. Caso se adicione mais algum tipo de informação ao *template* através da interface de *WebAdmin*, é preciso voltar a correr o *DCEGen*, para este juntar as alterações pedidas ao código.

Com a outra ferramenta, é possível sincronizar a base de dados com os comandos e a informação relativa ao sensor, presente nos ficheiros .cpp. Por exemplo, se há uma alteração na base de dados, é preciso correr o *sql2cpp* de novo e assim, a porção de código que for referente à versão anterior da base de dados já não é executada, mantendo, desta maneira, o código sempre actualizado e sincronizado.

Resta implementar agora as funções relativas aos comandos definidos previamente, escrevendo o código C++ adequado e compilá-lo para finalizar a criação do *template*.

Em apêndice (na parte A.2), é possível consultar uma descrição mais detalhada dos passos atrás referidos (instalação de um ambiente de desenvolvimento; utilização das ferramentas *DCEGen* e *sql2cpp* e ainda a compilação definitiva do *template* do dispositivo).

Por fim, deve incluir-se o sensor no sistema, adicionando um novo dispositivo baseado no *template* criado, acedendo a "*Advanced - Configuration - Devices*"na janela à direita no *WebAdmin*, e seleccionando o *device template*, anteriormente feito. Ainda nesta página, indica-se também o IP do sensor e, finalmente, acrescenta-se o mesmo.

Nota:
A informação recolhida nesta secção está presente de modo resumido numa página criada por mim, na *Wiki* do *LinuxMCE* e aprovada pelos *developers*.

3.4 Funções implementadas

As funções criadas foram essencialmente o programa de envio (servidor) e o de recepção (cliente) de informação dos sensores. Como já foi referido na secção da arquitectura, o lado do servidor corre na placa *OpenWrt*, onde também está presente o código do sensor em si. A parte do cliente reside no *Core PC*, ou seja, no *LinuxMCE* e ambos comunicam via *sockets*. A aprendizagem da configuração e programação destas entidades exigiu uma leitura prévia de [65] e [66].

De ambos os lados existem dois *buffers*, um para recepção e outro para envio, que actuam como mediadores dos dados dentro de cada programa. A nível geral, a informação, neles contida, é enviada pela rede como *strings*, quer seja a identificação por ID, quer seja a requisição dos comandos, ou até o resultado dos últimos. Cabe, depois, aos dois programas efectuarem as conversões do tipo de variável adequadas ao seu uso.

3.4.1 Servidor

O servidor baseia-se num modelo de programação simples, como se pode constatar na figura 3.8.

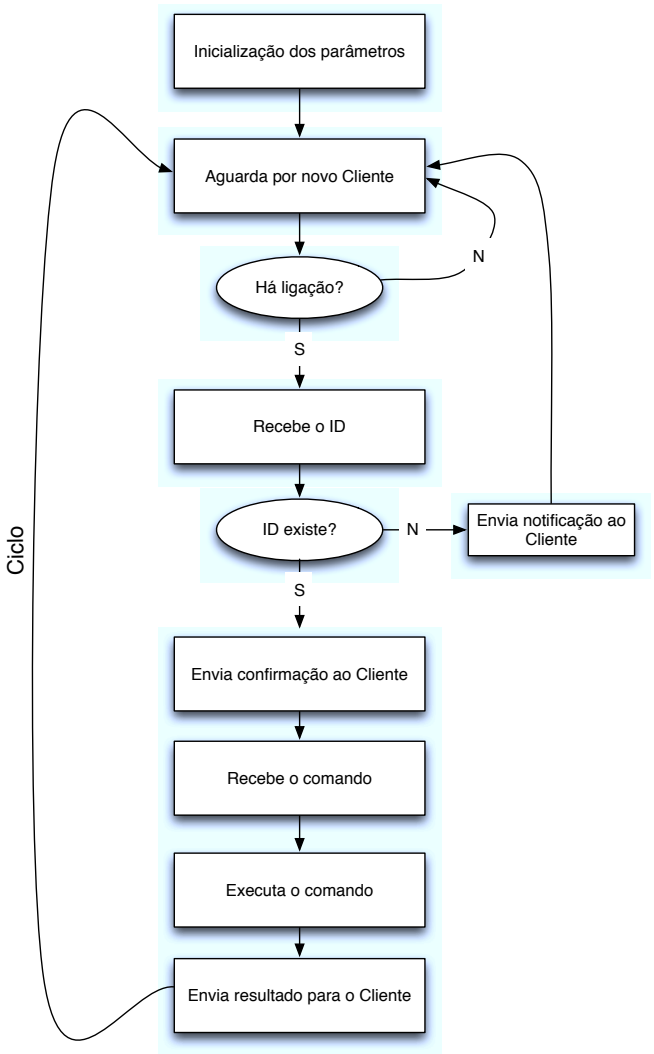

Figura 3.8: Diagrama de fluxo do Servidor

A nível funcional, o servidor pode ser descrito através dos seguintes pontos:

- Inicialmente é feita a configuração dos parâmetros específicos da conexão, como a porta onde o servidor irá correr, a inicialização das estruturas necessárias à criação do *socket* e consequente ligação;

- De seguida, o servidor entra em modo de espera, através dum ciclo infinito (*while(1)*), no qual aguarda a chegada duma conexão por parte do cliente. Caso essa ligação ocorra, é aceite, e é gerado um novo processo, através da função *fork()* que chama a função (*architecture()*) responsável pelo funcionamento da arquitectura descrita. Assim, dentro desta, há uma definição de cinco sensores de teste, cada um contendo informações diferentes (na variável *info*) e com o estado e a variável dinâmica a mudar constantemente para conferir um carácter real aos mesmos;

- É então feita a aquisição do ID do cliente conectado, a comparação dele com os existentes e, se o mesmo existir, é iniciado o próximo passo (caso contrário, o servidor envia uma notificação ao cliente que, neste momento, se desliga do sistema);

- Nesta próxima etapa, é recebido o comando desejado e, se este estiver presente no programa, é executado (adquirindo ou alterando o valor das variáveis da estrutura específica do sensor em questão) e o seu resultado enviado de volta para o cliente;

A simulação dos sensores é feita através da definição de um novo tipo de estrutura (*device*), constituída pelo ID do sensor, o seu estado, uma outra variável, que muda ao longo do programa (ilustrando a mudança de algum parâmetro do sensor) e uma *string* de informação genérica sobre ele.

De notar que o servidor suporta ligações simultâneas, podendo assim gerir situações como a conexão de mais do que um cliente, muito comuns num AP, que contenha vários sensores.

Para o iniciar, basta executá-lo com o número da porta onde se pretende que ele corra:

```
./server 1234
```

3.4.2 Cliente

Quanto ao cliente, descrito na figura 3.9, é na essência parecido com o servidor, mas tem como grande diferença o facto de, após obter o resultado do comando desejado, fechar a ligação aberta inicialmente.

Todo o código do cliente corre na função *main()* do programa, e a parte inicial é igual ao servidor. Em primeiro lugar, é realizada a conexão através da função *connect()* e enviado um pedido de confirmação da existência do ID deste sensor. Se a resposta vier negativa, o cliente aborta e gera uma notificação para o terminal; caso contrário, a execução prossegue e é enviado o comando desejado. Por fim, é recebido o resultado do comando, os dados são tratados, se assim for necessário, e a ligação existente terminada.

Na integração com o *LinuxMCE*, a parte de comunicação via *sockets* é igual, diferindo apenas no facto de, o pedaço de código referente à indicação do comando e à recepção do

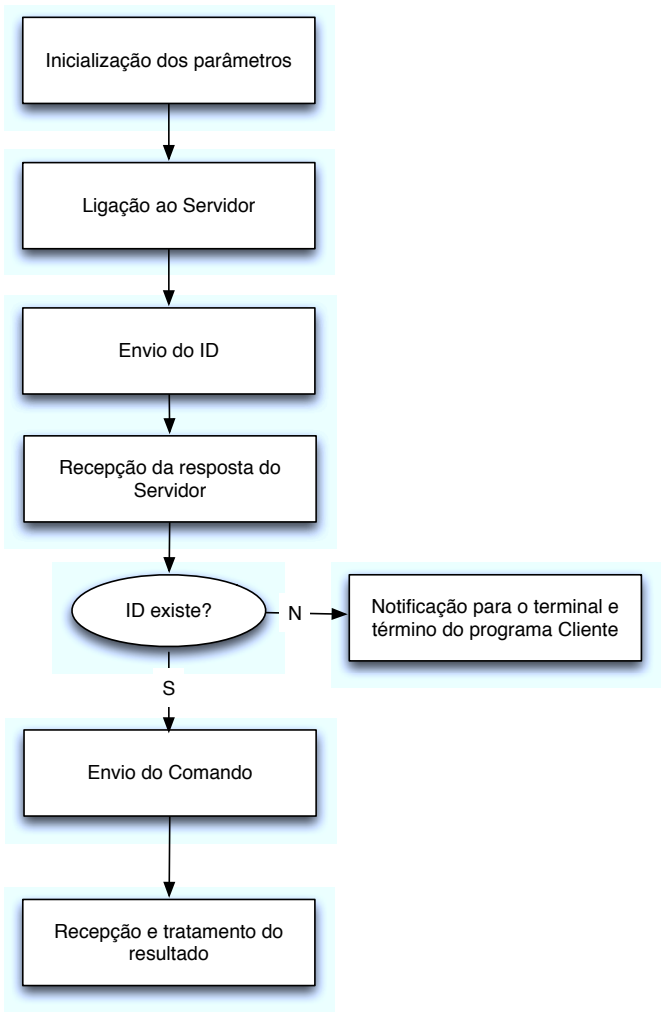

Figura 3.9: Diagrama de fluxo do Cliente

resultado do mesmo, ser colocada no esqueleto do *template* em cada comando desejado, como se pode verificar no código completo do modelo do sensor no apêndice B.1 (a preto está o esqueleto original do código e a azul um comando exemplo adicionado).

Além desta alteração, é adicionada também no *template*, a função específica do *LinuxMCE*, para enviar a informação do sensor requisitada. Esta é tratada do mesmo modo que todas as outras no sistema DCE (já explicado em 2.4.1.3) e, como tal, é executada com o envio do comando *SendPopupToAllOrbiters*, através do *SendMessage*. Os parâmetros de entrada neste caso são:

- a origem da mensagem (`m_dwPK_Device`, que se refere ao dispositivo representado por este *template*);
- o destinatário da mensagem (`DEVICETEMPLATE_VirtDev_All_Orbiters_CONST`, valor que corresponde a todos os *Orbiters* instalados no sistema);

- a prioridade (`PRIORITY_NORMAL`);
- o tipo de mensagem (`MESSAGETYPE_COMMAND`);
- o comando em questão (`COMMAND_Display_Alert_CONST`).

Para além destes parâmetros são também definidos alguns adicionais, específicos do comando:

- a frase a enviar no *popup* (`COMMANDPARAMETER_Text_CONST`);
- um *token* de controlo (`COMMANDPARAMETER_Tokens_CONST`, que pode ser qualquer texto);
- o tempo de permanência nos ecrãs do popup (`COMMANDPARAMETER_Timeout_CONST`);
- o valor da interrupção (`COMMANDPARAMETER_Interruption_CONST`, que se usa 0 neste caso).

Ao contrário do servidor, o cliente possui, dentro do programa, a porta à qual se deve ligar, para facilitar a integração com o *LinuxMCE*, podendo executá-lo somente com:

```
./client
```

Para pôr em prática este código, foi montado um demonstrador descrito no capítulo seguinte.

Capítulo 4

Demonstrador

4.1 Introdução

O demonstrador foi pensado com vista a ilustrar e pôr em prática todo o funcionamento do sistema tanto a nível de *LinuxMCE*, como da *gateway* entre este e os APs. Na figura 4.1 é possível observar o diagrama geral do demonstrador, contendo os componentes necessários ao *LinuxMCE* e os específicos a este projecto, bem como o tipo de ligações entre os AP's e o *Core PC*.

Através da *gateway* criada, que estabelece a ponte entre os dispositivos e a plataforma de gestão domótica, o *LinuxMCE*, pretende-se incorporar e posteriormente aceder às informações relativas aos dispositivos através de vários métodos de interacção (como *WebAdmin*, visualização em *floorplan*, e em *Orbiters*, tanto a nível local como remotamente). Além disso, também é necessário avaliar se o sistema funciona de forma satisfatória, nomeadamente a nível de resultados temporais.

4.2 Estrutura do Demonstrador

A nível macroscópico estão presentes, no demonstrador, vários componentes de hardware: o já conhecido *Core PC*, apresentado anteriormente, a *gateway* para a internet, os sensores e três placas da OpenRB[67].

Placas OpenRB

Como já foi referido, as placas correm *OpenWrt*, o que possibilita uma configuração à medida para o tipo de utilização pretendida em cada caso. A primeira delas (ligada directamente ao *Core PC* através da porta *eth1*) tem as funcionalidades de origem e actua como um *switch* que interliga o resto do sistema, em que a única particularidade alterada foi desactivar o servidor DHCP do *firmware* (os detalhes desta operação e da placa em geral estão presentes no apêndice C), porque o próprio *LinuxMCE* já fornece um servidor DHCP para utilizar na sua rede interna. Deste modo, evitam-se eventuais conflitos de

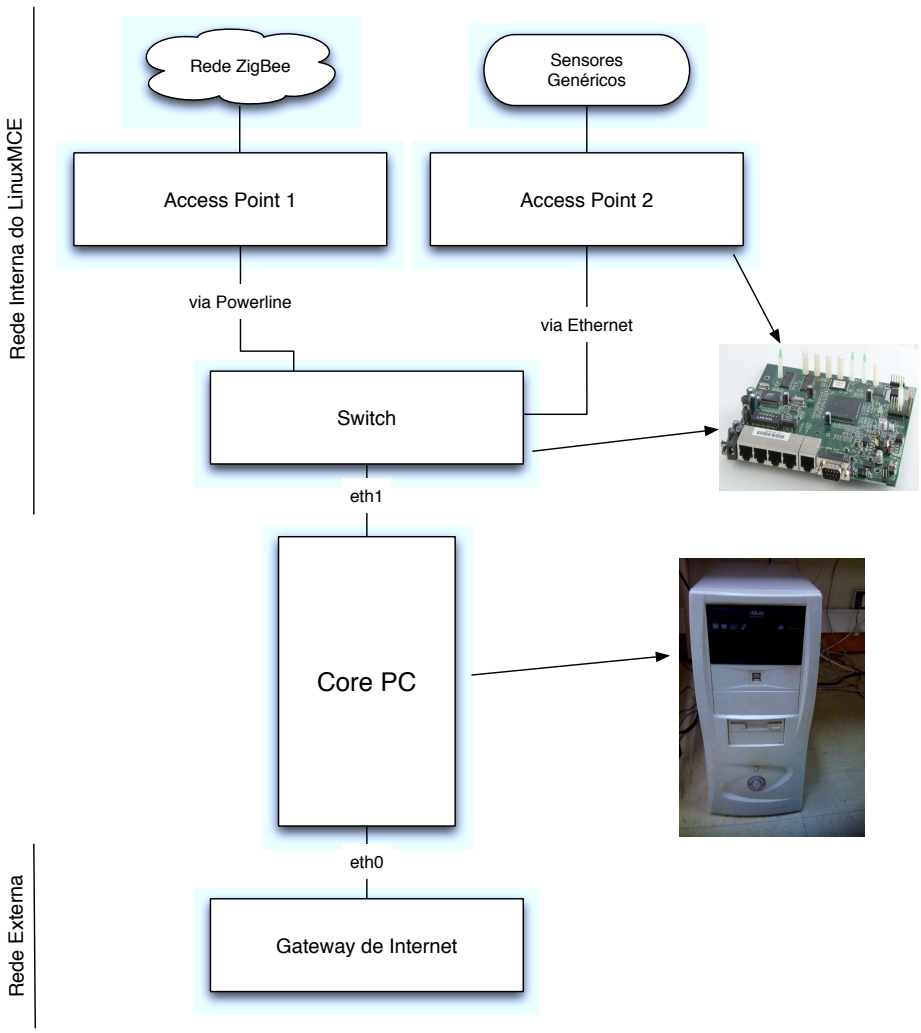

Figura 4.1: Diagrama geral do Demonstrador

atribuição de IPs a novas entidades que sejam ligadas na LAN.

As outras duas placas, uma ligada ao *switch* via *Ethernet* (AP2) e a outra através de *Powerline* (AP2), foram configuradas como APs e com o objectivo de albergarem e comunicarem com os sensores a elas conectados. Por actuarem como *Access Points*, estas placas permitem o acesso sem fios ao *LinuxMCE* para, por exemplo, aceder ao sistema em qualquer parte da casa, através dum *Orbiter* ou por *WebGUI* .

Ligações

Quanto a ligações, todo o demonstrador é interligado via *Ethernet*, à excepção do AP1 que estabelece a comunicação entre ele e o sistema pela rede eléctrica. A utilização deste tipo de meio foi pensada de modo a ilustrar que é possível estender facilmente o alcance do sistema ao resto da casa, incluindo outras divisões, mantendo o seu funcionamento

integral e usando diversas tecnologias de comunicação. Outro factor reside no facto de, apesar de existirem nos edifícios outras redes de transporte de informação, a rede eléctrica é omnipresente e pode ser apelativa nos casos em que não seja economica e/ou fisicamente viável a utilização de uma rede de dados cablada em determinados locais, e as redes não cabladas não sejam as mais adequadas.

A ligação é executada com um *Kit Powerline* da *Asoka* [68] , que contém um adaptador de rede eléctrica para *Ethernet*. Mais detalhes sobre a configuração e testes do *kit* podem ser consultados no apêndice D.

4.3 Interacção com o LinuxMCE

Depois do demonstrador configurado e montado, é necessário saber como iniciar os sensores e interagir com o sistema.

Além de se poder visualizar directamente a informação acerca dos sensores e da habitação através do *Core PC*, configurando um *Web Orbiter*, também é possível aceder à mesma em qualquer parte da casa, através de um computador ou dispositivo que possa ser ligado à rede interna do *LinuxMCE* ou por acesso remoto através de qualquer outro PC, mesmo da rede externa.

Para configurar um *Web Orbiter*, tem de se aceder à página de *Web Admin*, carregar em "*Show devices tree - CORE*"e, na página que aparece à direita, clicar em *Create Child Device*. É aberta uma nova *frame* (figura 4.2) em que, na parte da descrição, se insere o nome desejado para o novo *Orbiter* e, posteriormente, se selecciona *Pick device template*. Na janela que aparece, escolhe-se, como categoria do dispositivo (*Device Category*), periféricos (*Peripherals*) e na lista dos *Device Templates*, o *Generic Web Device*. Por fim, carrega-se ainda nesta janela, em *Pick device template*, o que faz com que ela se feche e o novo *Web Orbiter* seja adicionado ao sistema. Este processo ainda demora algum tempo e, por isso, deve-se esperar que o sistema exiba uma notificação que acabou de criar o dispositivo. Quando tal facto acontecer executa-se um "*Quick Reload Router*"(por exemplo através do *Orbiter* do *Core PC*, na parte *Advanced*) e está pronto a funcionar.

De modo a verificar que o *Web Orbiter* se encontra correctamente instalado, basta ir até "*Wizard - Devices - Orbiters*"(ilustrado na figura 4.3), onde se pode observar que estão já presentes dois *Orbiters*: o *Onscreen Orbiter* pertencente ao *Core PC* e um *Generic Proxy Orbiter* relativo ao último adicionado.

O acesso ao *Web Orbiter* é feito através dum *web browser* com a introdução do endereço `http://192.168.80.1/lmce-admin/weborbiter.php`. Tem de se efectuar a autenticação com o utilizador e senha de um dos habitantes da casa, seleccionar o nome do *Web Orbiter* criado e, por fim, o utilizador é presenteado com a sua interface (já ilustrada na figura 2.19).

Outro complemento interessante reside no facto de poder associar o sensor a uma divisão específica da casa, e posicioná-lo no sítio desejado, na planta dessa divisão. Para isso,

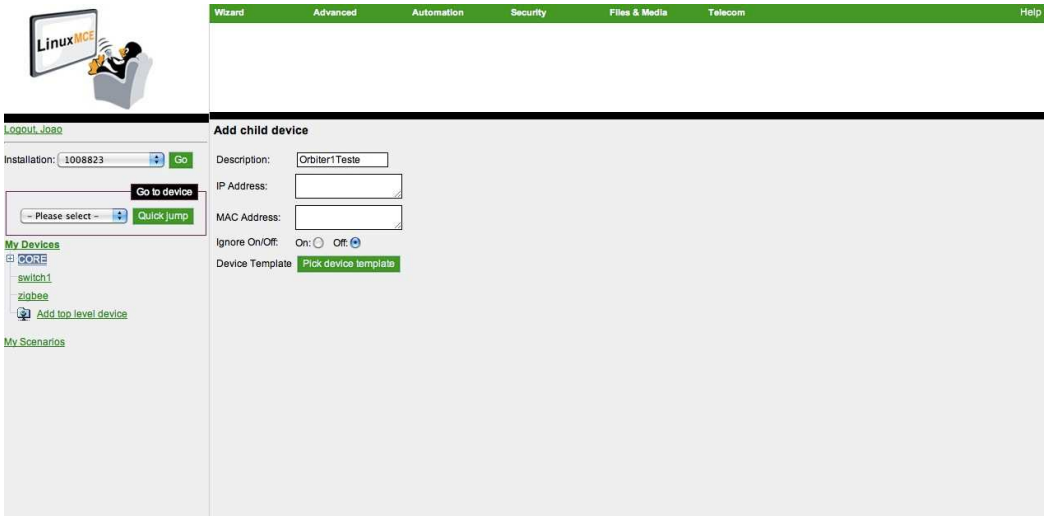

Figura 4.2: Ecrã de criação dum *Web Orbiter* no *LinuxMCE*

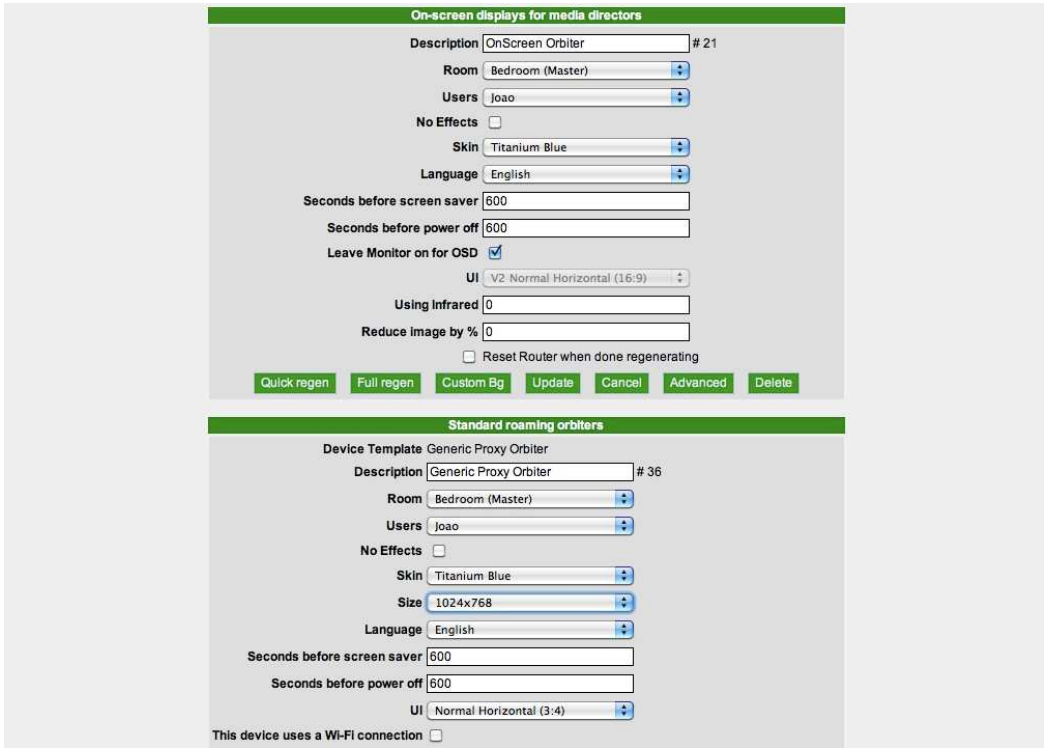

Figura 4.3: Ecrã de apresentação dos *Orbiters* no *LinuxMCE*

executa-se o conjunto de acções:

1. Acede-se através da *Web Admin*, no menu da esquerda, à categoria "*Devices*"
2. Clica-se em "*Floorplan Wizard*"

3. Carrega-se uma imagem com a planta da divisão

4. Selecciona-se o sensor em questão e arrasta-se o mesmo para a posição final na planta. (nas figuras 4.4 e 4.5 é ilustrado como é apresentada a página antes e depois de inserir o sensor na planta, respectivamente)

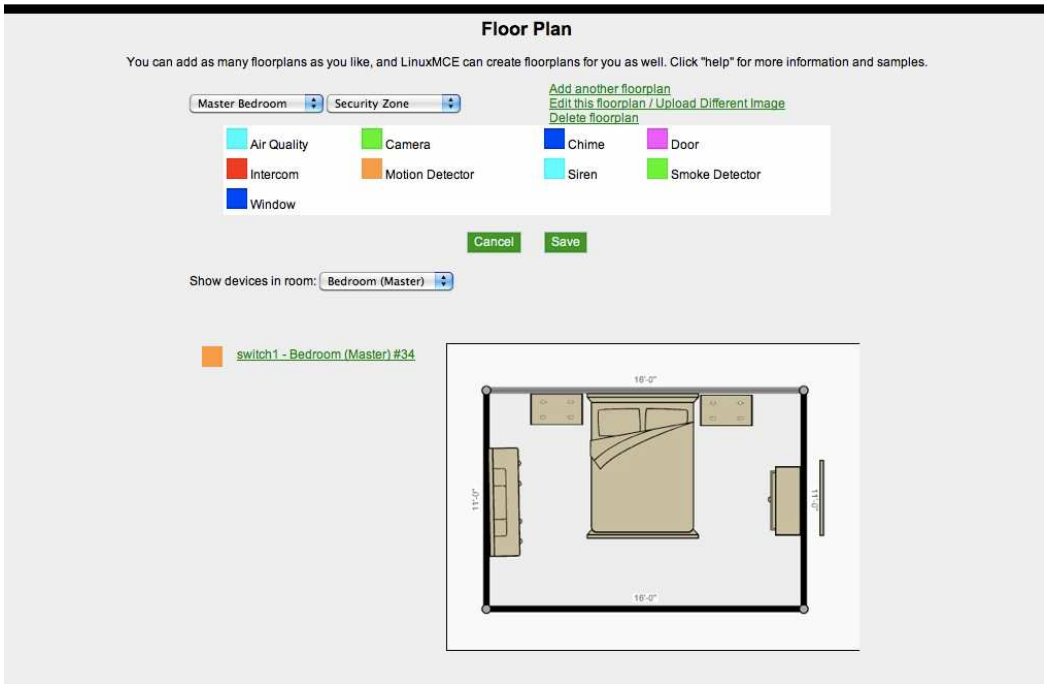

Figura 4.4: Ecrã de edição do *floorplan* no *LinuxMCE* antes de adicionar um sensor

Após configurados os meios necessários para visualizar informação sobre o sistema, resta iniciar o sensor através do conjunto de passos seguintes:

- Aceder à pasta do dispositivo:

```
cd /usr/src/lmce/LinuxMCE-0810/src/nomedodispositivo
```

- Executar o comando seguinte numa janela de terminal, no *Core PC*, para o iniciar:

```
./nomedodispositivo -r dcerouter -d devicenumber
```

O parâmetro *devicenumber* pode ser consultado através da interface de *WebAdmin*, no menu "*Advanced - Configuration - Devices*", seleccionando o dispositivo desejado e observando o número à frente de *Device Info*, como é apresentado na figura 4.6.

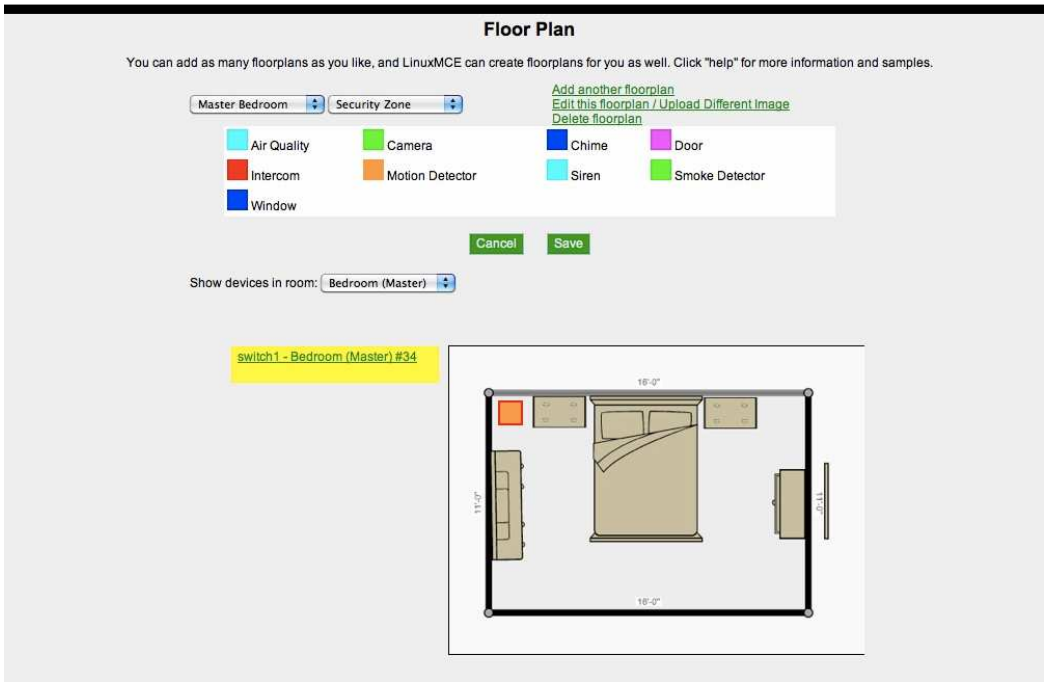

Figura 4.5: Ecrã de edição do *floorplan* no *LinuxMCE* depois de adicionar um sensor

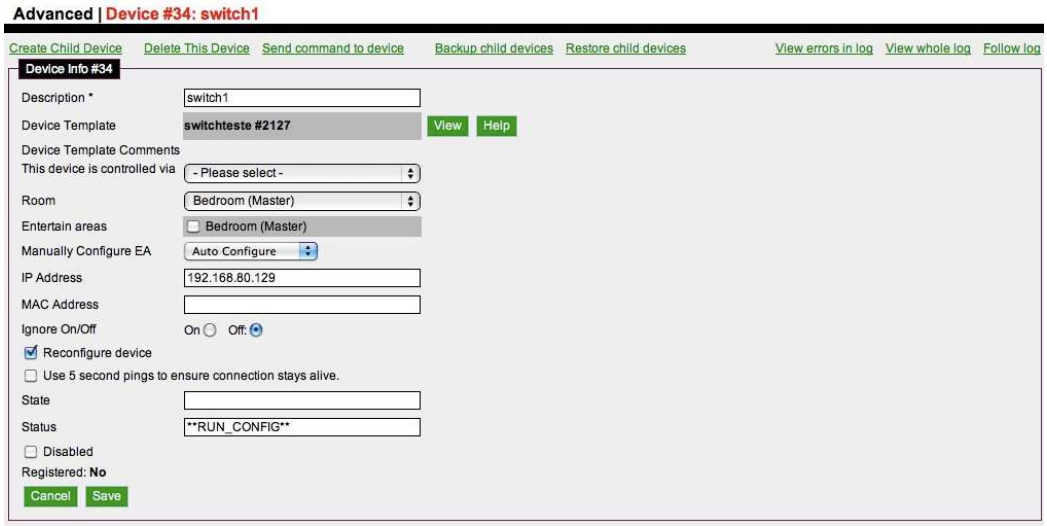

Figura 4.6: Ecrã de informação de um dispositivo no *LinuxMCE*

É de notar que pode estar a correr no sistema mais do que um sensor, como seria de esperar, já que o servidor, criado no capítulo 3, foi preparado para suportar de raiz esta situação.

A nível de interacção propriamente dita, do lado do *LinuxMCE*, existe a possibilidade de enviar os comandos previamente definidos no *template* para o sensor. Para isso, depois

de aceder via *Web Admin*, a "*Advanced - Configuration - Devices*", carrega-se no sensor e na *frame* da direita, em *Send Command to Device*. É aberta então uma nova janela (figura 4.7) que permite enviar um dos comandos presentes no modelo do sensor.

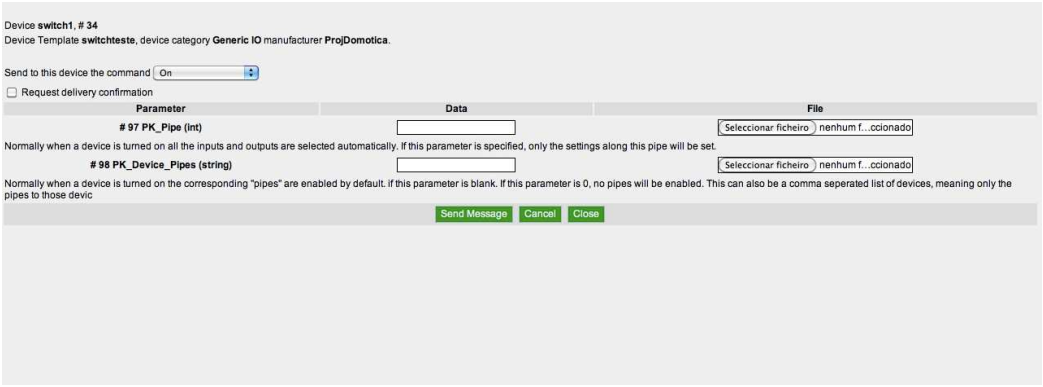

Figura 4.7: Ecrã referente ao envio de comandos no *LinuxMCE*

No caso do trabalho efectuado, como não foi implementado o sistema DCE nos dispositivos, não é necessário preencher os campos `PK_Pipe` e `PK_Device_Pipes`, sendo suficiente enviar o comando, que é executado segundo a porção de código que estiver definida no *template*.

Posteriormente à requisição do comando, espera-se uma resposta por parte do sensor. Como já foi explicado anteriormente, estas notificações são reportadas ao *LinuxMCE* sob a forma de *popups*, e podem ser visualizadas, tanto no *Orbiter* do *Core PC*, como no *Web Orbiter* criado no início desta secção (figura 4.8).

Finalmente, também se pode consultar a planta da divisão em que se encontra o sensor, nos dois ambientes referidos, na posição indicada anteriormente, aquando da colocação deste no *floorplan* (figura 4.9).

Caso seja pretendido, também é possível aceder à distribuição através de acesso remoto. Para isso, é preciso activar, no separador "*Wizard - Security - Outside Access*", a opção "*Allow outside access to the website on port 80*", como é ilustrado na figura 4.10.

Consegue aceder-se, então, a todo o conteúdo do *LinuxMCE*, tal como se estivesse a trabalhar na sua rede interna, como por exemplo entrar na *WebAdmin* através da inserção do endereço `http://IPexterno/lmce-admin`, ou no *WebOrbiter* criado, acessível no URL (*Uniform Resource Locator*) `http://IPexterno/lmce-admin/weborbiter.php`, e assim poder verificar o estado de toda a casa de forma transparente, até em locais exteriores à mesma.

É evidente que, ao expor todo o controlo de uma casa ao exterior, é preciso ter consciência que, qualquer falha de segurança que possa permitir o acesso à rede, pode ser fatal para a integridade do sistema. No caso de uma instalação de domótica, como é

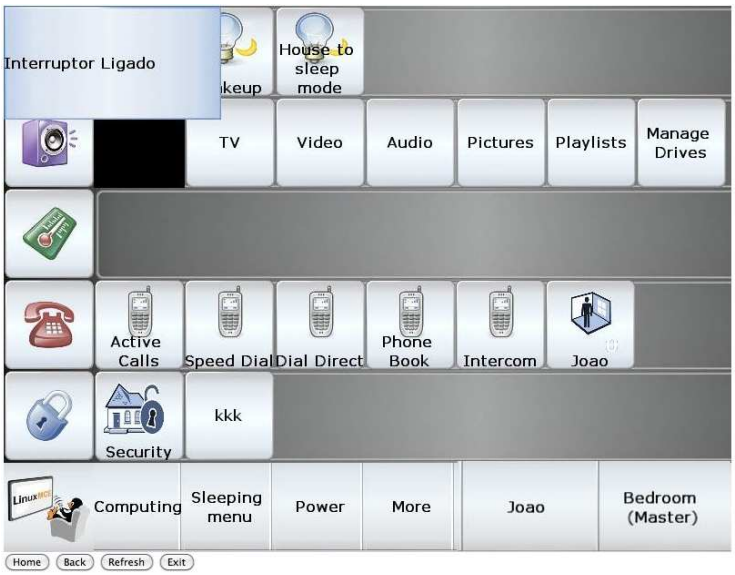

Figura 4.8: Ecrã da recepção dum *popup* num *Web Orbiter* no *LinuxMCE*

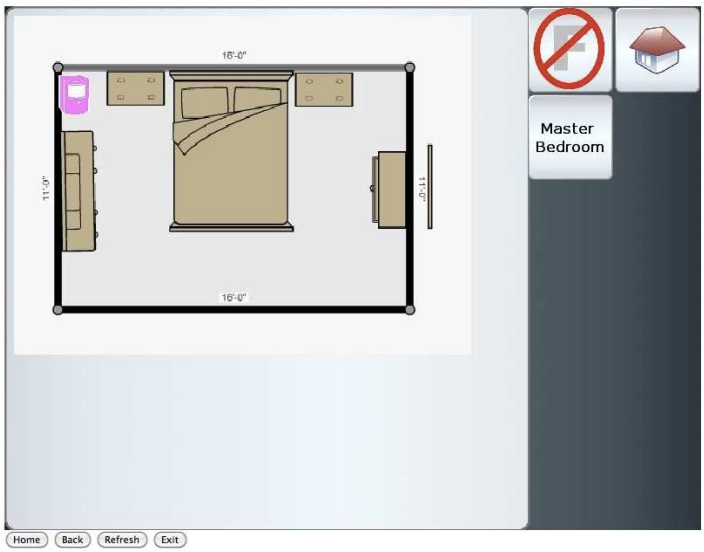

Figura 4.9: Ecrã de visualização do *floorplan* num *Web Orbiter* no *LinuxMCE*

fácil imaginar, isso implica poder controlar actuadores vitais ao funcionamento da casa, os quais podem pôr em perigo a segurança dos seus habitantes. Para colmatar esta falha ou, pelo menos, reduzir a possibilidade de intrusão no sistema, pode adquirir-se um certificado SSL (*Secure Sockets Layer*) de uma empresa de segurança como a *Verisign* [69], embora todo o processo possa atingir custos elevados.

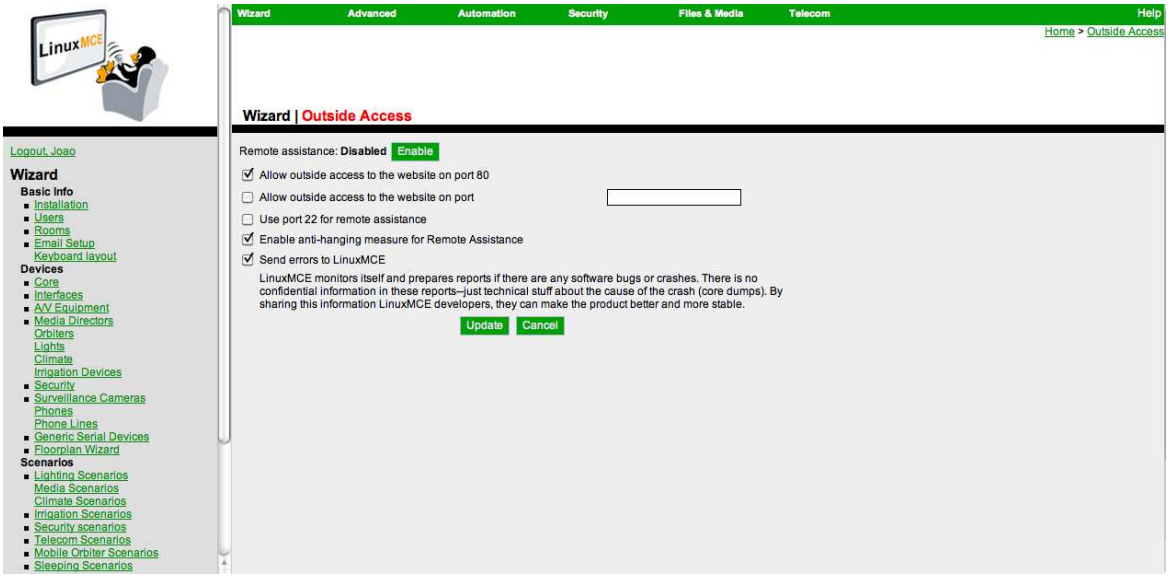

Figura 4.10: Ecrã de configuração do acesso remoto no *LinuxMCE*

4.4 Resultados

A nível de funcionamento geral do sistema, foi possível comprovar que ele se comporta adequadamente em relação aos objectivos iniciais do projecto, que passavam por aceder e visualizar no *LinuxMCE*, a informação de dispositivos presentes em placas com *OpenWrt*.

Para uma análise mais cuidada, em termos de resposta temporal do sistema como um todo, foram feitas várias experiências. Depois do demonstrador estar completamente montado e a funcionar, inseriu-se código na parte do cliente (a verde no apêndice B.2), portanto do lado do *LinuxMCE*), de um dispositivo, de modo a medir o tempo gasto entre a requisição e visualização de informação do mesmo. O método usado baseou-se na utilização da função *gettimeofday()* do *Linux*, que assegura uma precisão na ordem dos milissegundos. Para assegurar a fiabilidade dos resultados foram efectuadas 1000 medidas para cada uma das situações, recorrendo a um *script* criado, que envia o pedido dum comando mil vezes seguidas.

Numa primeira experiência, apontaram-se os valores referentes ao envio de três comandos diferentes para o dispositivo, como se pode ver na tabela ilustrada pela figura 4.11, recorrendo apenas a ligações *Ethernet* entre todo o sistema.

Posteriormente, a conexão entre o *Core PC* e o AP foi substituída por uma ligação *powerline* para tentar perceber se a rede eléctrica seria muito limitativa neste cenário. Estas medidas estão apresentadas na tabela representada pela figura 4.12.

Da análise entre as duas tabelas, podemos inferir que, apesar da rede eléctrica introduzir algum atraso, como seria de esperar, não é de modo algum um factor limitativo para o caso em estudo. O sistema continua a ter uma resposta bastante rápida (na ordem dos

	Mínimo (ms)	Máximo (ms)	Média (ms)
GetState	4.1210	143.8470	9.7132
GetDyn	3.4700	115.0800	6.6391
GetInfo	3.7910	130.0800	8.9406
Todos	3.4700	143.8470	8.4310

Figura 4.11: Tabela comparativa das medições temporais dos comandos sem *powerline*

	Mínimo (ms)	Máximo (ms)	Média (ms)
GetState	9.9300	146.3700	16.6784
GetDyn	9.1110	131.5360	13.2529
GetInfo	9.7420	131.6430	13.3484
Todos	9.1110	146.3700	14.4266

Figura 4.12: Tabela comparativa das medições temporais dos comandos com *powerline*

8.4 ms), comparativamente à situação em que toda a rede estava coberta por *Ethernet* (na ordem dos 14.4 ms). Em relação aos vários comandos, todos se comportam de forma relativamente semelhante pois não exigem grande poder/tempo de processamento.

Os histogramas apresentados abaixo foram o recurso utilizado para melhor ilustrar todas as experiências. As medições temporais sem recorrer à utilização de *powerline* estão presentes na figura 4.13, e a experiência com a *powerline* integrada na rede é representada pela figura 4.14.

Os valores fora da média podem explicar-se, porque o *Linux* executa o escalonamento dos processos de forma não linear durante o decorrer do tempo. Como tal poderia ocorrer um atraso na execução dos comandos, em determinados momentos em que existisse uma maior carga no processador. O facto da função *GetDyn* ser recorrentemente mais rápida é justificado apenas por ser aquela que necessita de menor poder de processamento.

Comparando os casos da inserção de *powerline* na rede e sem este, é possível dizer que existe uma diferença de sensivelmente o dobro do tempo para as medidas mais comuns, factor que não é de forma alguma limitativo para este trabalho, como já foi referido.

Tendo em conta todos os resultados, o sistema comporta-se de forma aceitável suportando sem problemas, vários pedidos em catadupa sem se deteriorar a nível temporal, e em média com uma capacidade de reposta muito rápida.

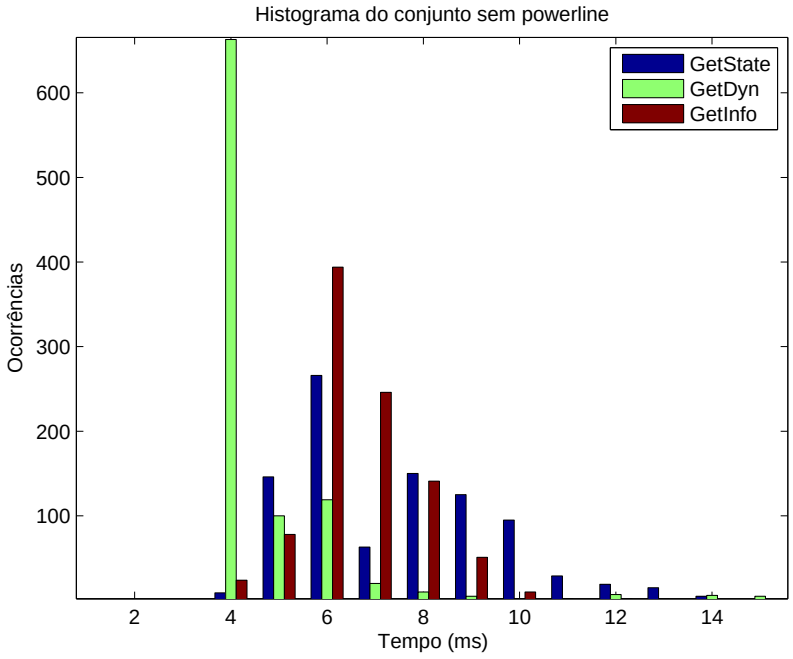

Figura 4.13: Histograma das medições temporais dos comandos sem *powerline*

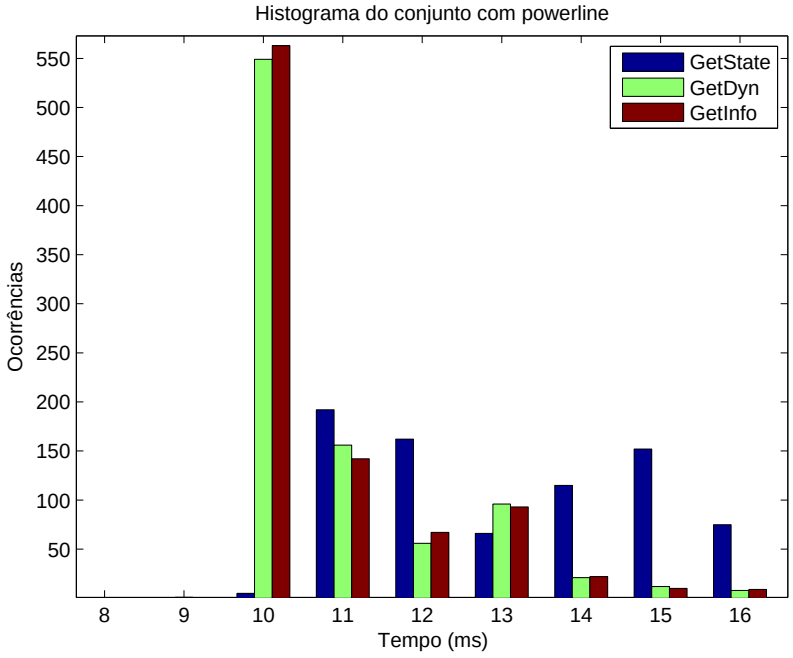

Figura 4.14: Histograma das medições temporais dos comandos com *powerline*

Capítulo 5

Conclusões

5.1 Resumo do trabalho realizado

O trabalho desenvolvido, nesta tese, focou-se na integração de dispositivos a correr sobre APs numa plataforma de gestão de domótica baseada em *Linux*, o *LinuxMCE* e posterior visualização do estado dos dispositivos através de vários formas de interacção. A ideia surgiu, por um lado, devido à quantidade de soluções e normas existentes, sem existir, no entanto, uma uniformização entre elas, oferecendo por isso aos interessados apenas soluções dispendiosas e limitadas. Por outro lado, a parca utilização das redes já existentes na maior parte das habitações (tanto cabladas como sem fios), fomentou a sua adopção para suporte deste trabalho.

Inicialmente, efectuou-se um estudo da distribuição em questão e foram adquiridos os métodos necessários para a sua instalação e configuração inicial. A etapa seguinte consistiu na criação de um modelo do dispositivo a adicionar ao sistema e em perceber qual a abordagem que deveria ser seguida para criar uma comunicação fiável entre os dispositivos e o *LinuxMCE*. Decidiu-se, então programar em C++, um cliente e um servidor genéricos baseados em *sockets*, contendo métodos de identificação dos dispositivos (ID), envio e recepção de comandos. O servidor corre nos APs que contêm os sensores e o cliente foi, posteriormente, integrado em cada *template* do lado do *LinuxMCE*.

Quanto a resultados, a *gateway* actua como um todo e possibilita, assim, a simbiose adequada entre os sensores e o *LinuxMCE*. A arquitectura de troca de mensagens criada funciona perfeitamente dentro dos requisitos (possuindo uma boa resposta também a nível temporal), permitindo estabelecer uma comunicação robusta entre os dois meios. Relativamente à interacção com o sistema, é possível enviar comandos a partir do *LinuxMCE* e receber a resposta proveniente dos sensores, através de um *popup* que aparece em todos os ecrãs (*Orbiters*) instalados. A nível de informação, está disponível também a localização dos dispositivos numa planta da divisão da habitação, à qual se pode aceder igualmente através de qualquer PC conectado à LAN interna do sistema.

Um factor limitativo do trabalho efectuado prendeu-se com a falta de integração do sistema de mensagens DCE e, assim, não foi possível atingir um maior nível de interacção com

os sensores a partir do *LinuxMCE*, bem como receber eventos accionados pelos dispositivos.

5.2 Trabalho futuro

Tendo em conta o trabalho realizado, ainda existem alguns aspectos que podem ser desenvolvidos e melhorados em estudos futuros.

Implementação do sistema DCE

Como foi referido na secção anterior, é necessário existir uma maior interacção com os sensores a partir da distribuição utilizada e, por isso, poderá optar-se por implementar o sistema de mensagens do *LinuxMCE* nos próprios APs, como é ilustrado na figura 5.1. Tal facto permitiria que pudessem ser recebidos comandos e eventos com um maior nível de minúcia no sistema, apresentando a informação daí proveniente, não só via *popups* mas também integrada na planta de cada divisão. Com esta evolução, seria possível controlar os sensores e, portanto, enviar comandos e receber eventos, directamente na *floorplan*. No entanto, a abordagem utilizada neste trabalho de dissertação tem a vantagem de permitir incorporar o código relativo aos dispositivos em equipamentos com especificações mais modestas, o que não aconteceria se o sistema DCE estivesse implementado, pois este necessita dum maior poder de processamento.

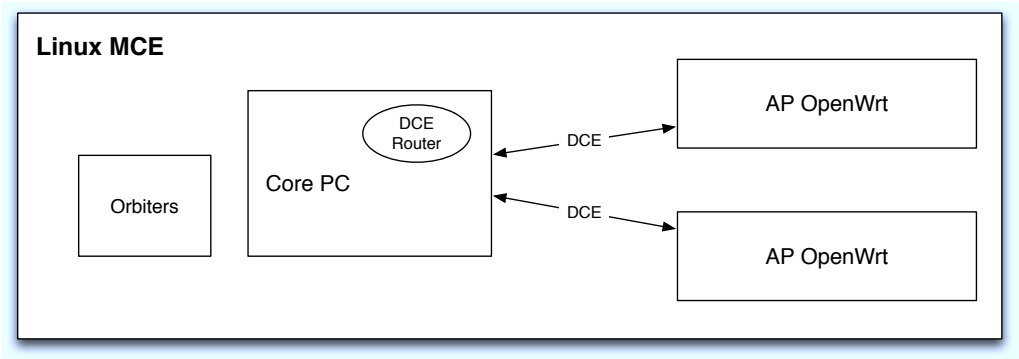

Figura 5.1: Diagrama do sistema com as mensagens DCE incorporadas

Criação de Eventos

Outro ponto interessante a adicionar, seria dotar os sensores de capacidades para enviar eventos *time-triggered* ou *action-triggered*. Para esse efeito, deveria adaptar-se o código dos sensores, de modo a reportarem essa informação que, em conjunto com o sistema de mensagens DCE, remeteria para o *LinuxMCE* o tratamento e análise desses mesmos eventos e permitiria criar cenários mais avançados de controlo e segurança, com a facilidade da interface de administração *Web GUI* da distribuição.

Apêndice A

Instalação e Configuração do LinuxMCE

A.1 Geral

Antes de iniciar a programação da *gateway*, foi preciso instalar e configurar minimamente o *LinuxMCE*. Este processo é ilustrado detalhadamente neste apêndice, de modo a solidificar o factor de reprodutibilidade do trabalho.

Para começar, é imperativo efectuar a descarga da versão 8.10 do *Kubuntu* e gravá-la em CD (Compact Disc). Quanto à instalação em si, existem dois métodos, mais ou menos semelhantes para a efectuar: um modo automático e um semi-automático.

Dado que, no momento em que esta tese foi escrita, o modo automático não produzia os efeitos desejados, seguiu-se o método semi-automático:

1. Inserir o CD na drive e entrar no menu da BIOS (*Basic Input/Output System*), para activar como primeiro dispositivo de arranque (*First boot device*) o leitor de CDs;

2. Quando o CD carregar, seguir os passos do ecrã, para proceder à instalação normal do *Kubuntu*;

3. Depois de instalada esta distribuição, é preciso abrir uma janela de terminal e executar a seguinte sequência de comandos, que descarregará a última versão do instalador do *LinuxMCE* e o instalará,[1]:

```
sudo su
apt-get update
sudo apt-get dist-upgrade
wget -c http://deb.linuxmce.org/ubuntu/new-installer-latest.tar.gz
tar xvf new-installer-latest.tar.gz
```

[1]Nota: É aconselhável correr os vários comandos (incluindo os *scripts* .sh), de forma faseada e não em catadupa, executando o código todo de uma só vez

```
cd new-installer
./pre-install-from-repo.sh
./mce-install.sh
./post-install.sh
```

4. Por fim reinicia-se o computador e passa-se à parte da configuração.

> **Nota:**
> Este método configura automaticamente o *Core PC* em modo híbrido, permitindo assim actuar sobre o sistema com apenas um computador, pois tem-se acesso às funcionalidades dum MD, sendo a mais relevante, neste caso, o *Orbiter* integrado.

De modo a não danificar a instalação, não se deve voltar a executar o comando *"sudo apt-get dist-upgrade"*, pois a versão actual do *LinuxMCE* está preparada e compilada apenas para correr em cima do *Kubuntu* 8.10.

Finda esta parte e depois de reiniciado o computador, a tarefa seguinte é a de configurar o sistema. A configuração é efectuada em duas fases, o *A/V Wizard* e o *House Wizard*.

A primeira é o *A/V Wizard* e apresenta-se inicialmente com o ecrã representado na figura A.1.

Figura A.1: Ecrã inicial do *A/V Wizard* no *LinuxMCE* [70]

De seguida (figura A.2), escolhe-se através de que ligação está conectado o ecrã usado no PC, a sua resolução, taxa de refrescamento (se aplicável) e confirmam-se as definições escolhidas.

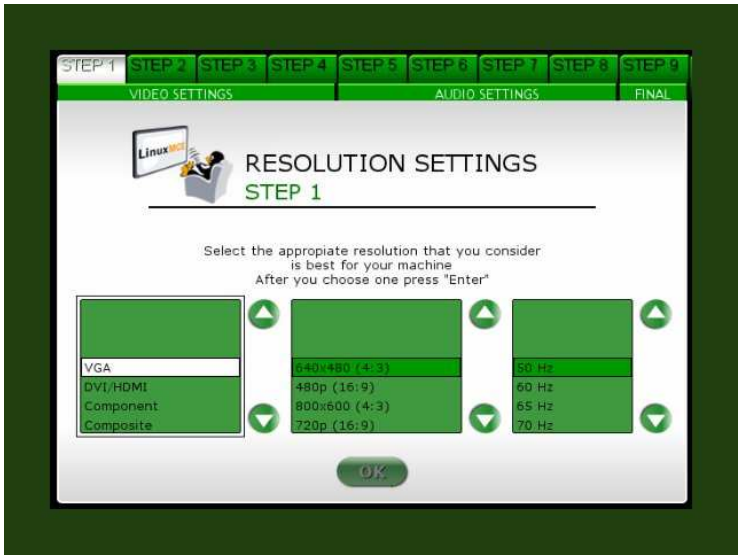

Figura A.2: Ecrã de escolha das resoluções no *A/V Wizard* do *LinuxMCE* [71]

No menu seguinte (figura A.3), é possível definir qual o aspecto gráfico da interface de utilizador e, de entre as opções disponíveis, deve seleccionar-se a adequada à placa gráfica, presente no PC. A primeira escolha *"Static images, no overlay"* funciona em praticamente todas as placas gráficas, pois não necessita de um grande poder de processamento, já que as imagens são todas estáticas e não é mostrado nenhum conteúdo multimédia[2]. A segunda já apresenta conteúdo multimédia, os menus são dinâmicos e, como tal, necessita de uma gráfica com suporte para *OpenGL*. Para utilizar a última opção, é necessária uma placa da *nVidia* das séries 6xxx, 7xxx, 8xxx, 9xxx e, possivelmente, superiores, com pelo menos 128MB de RAM (Random-Access Memory). Como bónus, este modo confere uma interface com transições suaves e efeitos mais avançados, como a reprodução dos conteúdos sempre em ecrã inteiro e, quando não estiver a ser visualizado vídeo, o sistema descarrega fotos do sítio online Flickr e apresenta-as em *slideshow*.

De seguida são apresentados mais alguns ecrãs, que permitem ajustar o tamanho da imagem em relação ao monitor do PC, seleccionar o tipo de ligação de som a utilizar (figura A.4), o volume, os testes de som , *Dolby* e DTS (*Digital Theater Systems*) e, por fim, um outro que apresenta o resumo das escolhas efectuadas (figura A.5).

Depois de seleccionar a opção "*I Agree*"é dada por concluída a parte de configuração das componentes de som e vídeo do PC, e entramos no segundo *wizard* referente à habitação propriamente dita (figura A.6).

Num primeiro passo, é preciso definir os nomes dos habitantes da casa que vão usar o

[2]Tem um aspecto igual ao de um *WebOrbiter* 2.19

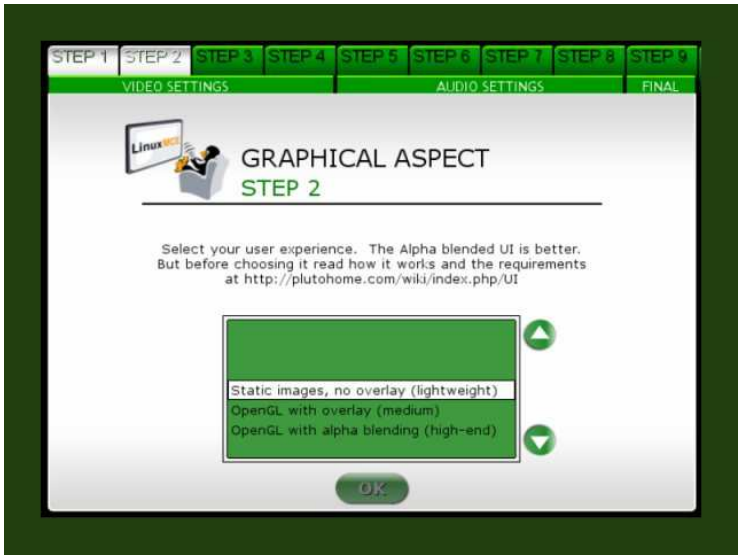

Figura A.3: Ecrã de escolha do aspecto gráfico no *A/V Wizard* do *LinuxMCE* [72]

Figura A.4: Ecrã de selecção da ligação de som no *A/V Wizard* do *LinuxMCE* [73]

sistema (figura A.7) e responder a algumas questões sobre a localização onde se encontra a habitação.

Posteriormente, estipula-se o tipo e quantidade de divisões da casa (figura A.8), e se há interfaces de luzes, sistemas de alarme ou serviços VoIP (no caso de não haver, basta seleccionar *"Continue without one"* e *"Next"*).

Para terminar, o utilizador é informado que, para adicionar uma planta de cada divisão

Figura A.5: Ecrã de resumo final do *A/V Wizard* no *LinuxMCE* [74]

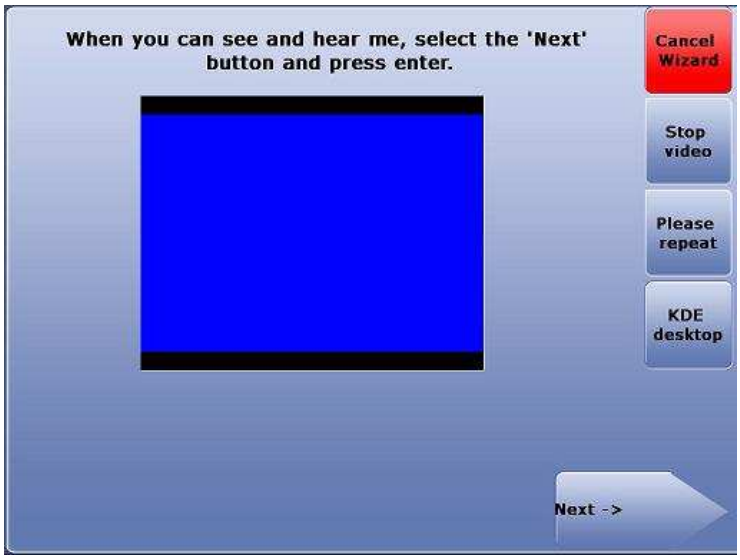

Figura A.6: Ecrã inicial do *House Wizard* no *LinuxMCE* [75]

da habitação, deve fazê-lo, posteriormente, através da interface de *WebAdmin*. Ao carregar em *"Next"* conclui-se a configuração da habitação no *House Wizard* (figura A.9)[3].

Este último ecrã tem como opção seguinte iniciar o *Media Wizard*, que ainda faz parte do *House Wizard*, como se pode ver pela semelhança dos menus. Para a instalação efectuada no âmbito da tese, não é necessário definir muitas opções neste ecrã, bastando seleccionar a sala em que estamos, como é pedido, e ir respondendo negativamente às questões que sur-

[3]É possível mais tarde, mudar as definições escolhidas

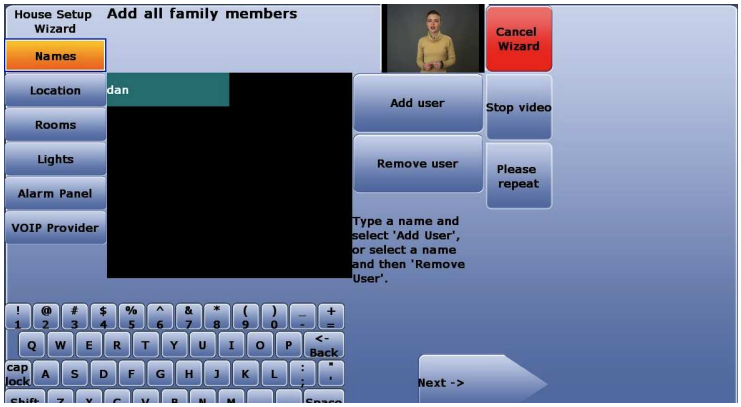

Figura A.7: Ecrã de definição dos utilizadores da casa no *House Wizard* do *LinuxMCE* [76]

Figura A.8: Ecrã de selecção das divisões da casa no *House Wizard* do *LinuxMCE* [77]

gem relacionadas basicamente com selecção de equipamento audiovisual, como televisões, amplificadores e outros.

Finalmente, é apresentada uma última imagem, na qual se deve clicar na opção *"Start using the system"*, o que fará com que o *DCERouter* seja reiniciado e o sistema iniciado com as configurações desejadas.

A partir daqui, o utilizador é presenteado com o ecrã inicial dos MDs representado na figura 2.18, e deve ir a "*Advanced - KDE Desktop*", abrir uma página *Web* no endereço `http://dcerouter` ou aceder ao mesmo, através de outro PC na rede interna do *LinuxMCE* [4], para definir o último aspecto essencial do sistema: as definições de rede. Depois de

[4]ou ainda em " *(Advanced - Computing - WebAdmin)*"de modo a não sair da interface do *LinuxMCE*

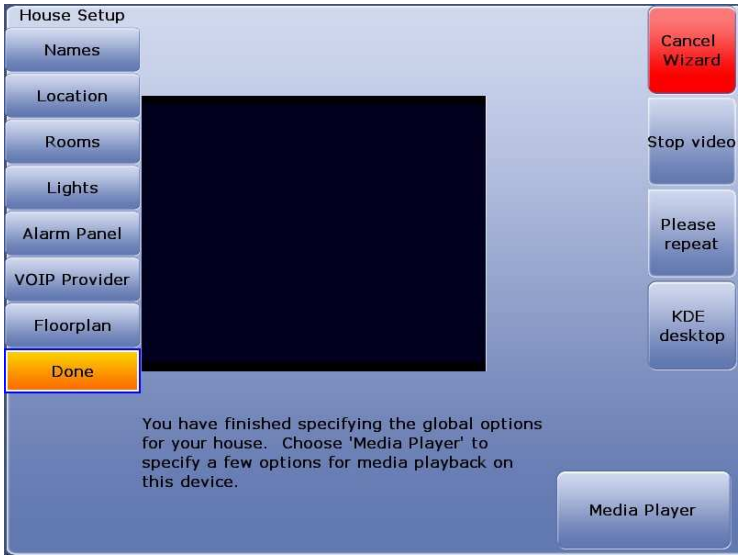

Figura A.9: Ecrã do resumo final do *House Wizard* no *LinuxMCE* [78]

efectuada a autenticação, é preciso carregar em "*Advanced - Network - Network Settings*"na *frame* principal e na página semelhante à figura A.10, verificar se a interface à qual está ligada a *Internet* e a rede interna do sistema estão correctas. Se não estiverem, basta clicar em "*Swap Interfaces*"para efectuar a troca.

> **Nota:**
> Para voltar ao *LinuxMCE* é suficiente aceder ao ambiente de trabalho 5, no *Kubuntu*.

É necessário ainda ter em atenção que, depois de cada alteração, é aconselhado fazer um "*Quick Reload Router*"e, se esta implicar diferenças nos ecrãs dos *Orbiters* (como mudança da posição dos sensores na planta ou modificação de algum outro aspecto gráfico), um "*Regen All Orbiters*", a partir do botão *Advanced* na interface principal.

A.2 Configuração e criação de templates e novos dispositivos

A instalação do ambiente de desenvolvimento no *Core PC* pode ser atingido com os seguintes passos, numa janela de terminal:

1. Instalar os pacotes essenciais para a configuração deste ambiente:

```
sudo apt-get install pluto-dcegen pluto-sql2cpp build-essential subversion
```

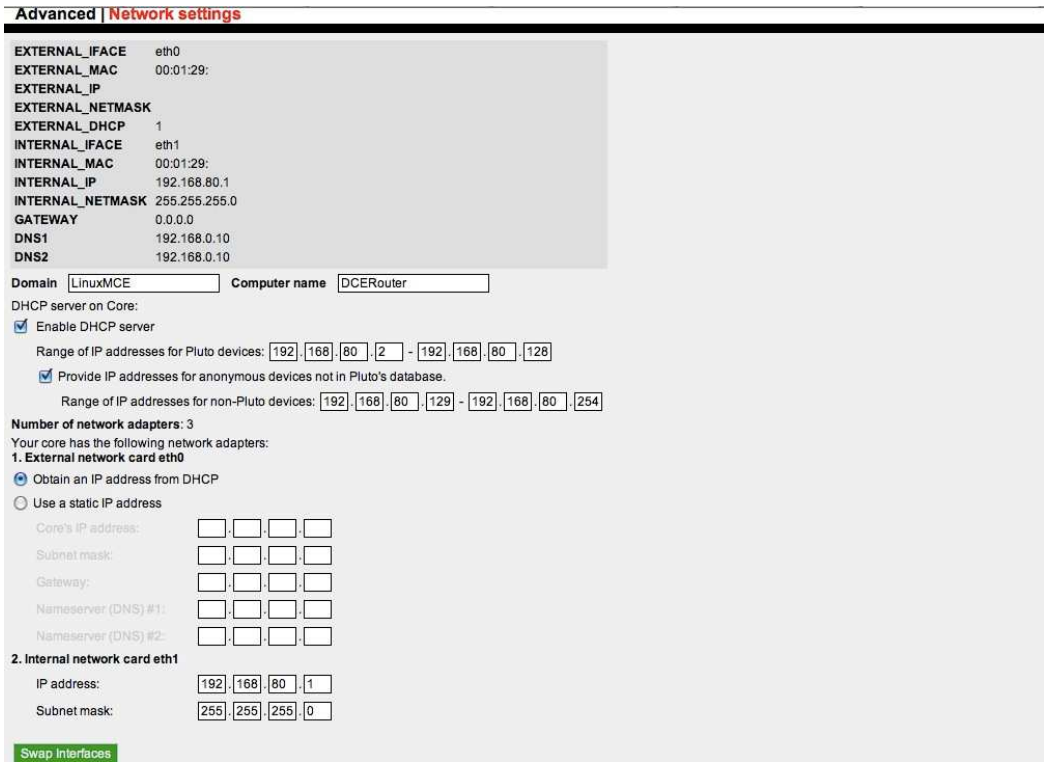

Figura A.10: Página de definições de rede do *LinuxMCE*

```
libmysqlclient15-dev
```

2. Criar uma pasta para onde se irão descarregar os ficheiros e atribuir as permissões correctas ao utilizador (substituir o parâmetro "seututilizador") do sistema:

```
cd /usr/src
sudo mkdir lmce
sudo chown seutilizador lmce
cd lmce
```

3. Efectuar a descarga dos ficheiros:

```
svn co http://svn.linuxmce.org/svn/branches/LinuxMCE-0810/
```

4. Copiar as bibliotecas da nossa instalação para o ambiente de desenvolvimento:

```
cp /usr/pluto/lib/* /usr/src/lmce/LinuxMCE-0810/src/lib
```

5. Exportar as variáveis de compilação seguintes:

```
export SNR_LDFLAGS="  "
export SNR_CPPFLAGS="-DKDE-LinuxMCE"
```

O passo 4 pode ser substituído, colocando as variáveis e valores indicados no ficheiro *.profile* do *Kubuntu*. Caso isto não seja feito, cada vez que o *Core PC* for reiniciado, é preciso voltar a exportá-las.

Para executar as ferramentas *DCEGen* e *sql2cpp* os passos são os seguintes:

1. Aceder à pasta onde estão as ferramentas:

   ```
   cd /usr/src/lmce/LinuxMCE-0810/src/
   ```

2. Executar o *DCEGen* com o parâmetro "*numerodotemplate*"que pode ser consultado na página de edição do *template*:

   ```
   DCEGen -d numerodotemplate
   ```

3. Executar o *sql2cpp*, que não necessita de argumentos, na pasta do modelo:

   ```
   cd /usr/src/lmce/LinuxMCE-0810/src/nomedodispositivo
   sql2cpp
   ```

Por fim, para compilar efectivamente o código criado para o novo dispositivo basta executar as instruções:

1. Aceder à pasta onde se localiza o modelo:

   ```
   cd /usr/src/lmce/LinuxMCE-0810/src/nomedodispositivo
   ```

2. Atribuir permissões de execução ao script `post_make.sh`:

   ```
   sudo chmod +x post_make.sh
   ```

3. Executar o comando seguinte para proceder efectivamente à compilação:

   ```
   make bin
   ```

4. Copiar o dispositivo para a pasta indicada (este passo só precisa de ser efectuado uma vez):

   ```
   cp /src/nomedodispositivo /usr/pluto/bin
   ```

Apêndice B

Código do Template LinuxMCE

B.1 Cliente

```
/*
     Copyright (C) 2004 Pluto, Inc., a Florida Corporation

     www.plutohome.com

     Phone: +1 (877) 758-8648


     This program is free software; you can redistribute it and/or modify it
         under the terms of the GNU General Public License.
     This program is distributed in the hope that it will be useful, but
        WITHOUT ANY WARRANTY; without even the implied warranty
     of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

     See the GNU General Public License for more details.

*/
//<-dceag-d-b->
#include "switchteste.h"
#include "DCE/Logger.h"
#include "PlutoUtils/FileUtils.h"
#include "PlutoUtils/StringUtils.h"
#include "PlutoUtils/Other.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#include <iostream>
```

```
using namespace std;
using namespace DCE;

#include "Gen_Devices/AllCommandsRequests.h"
#include "pluto_main/Define_DeviceTemplate.h"
#include "pluto_main/Define_Event.h"
#include "pluto_main/Define_EventParameter.h"
#include "pluto_main/Define_DeviceData.h"
#include "pluto_main/Define_Command.h"
#include "pluto_main/Define_CommandParameter.h"

#define MAXDATASIZE 100


//<-dceag-d-e->

//<-dceag-const-b->

switchteste::switchteste(int DeviceID, string ServerAddress,bool
    bConnectEventHandler,bool bLocalMode,class Router *pRouter)
  : switchteste_Command(DeviceID, ServerAddress,bConnectEventHandler,
      bLocalMode,pRouter)
//<-dceag-const-e->
{
}

//<-dceag-const2-b->
switchteste::switchteste(Command_Impl *pPrimaryDeviceCommand,
    DeviceData_Impl *pData, Event_Impl *pEvent, Router *pRouter)
  : switchteste_Command(pPrimaryDeviceCommand, pData, pEvent, pRouter)
//<-dceag-const2-e->
{
}

//<-dceag-dest-b->
switchteste::~switchteste()
//<-dceag-dest-e->
{
}

//<-dceag-getconfig-b->
bool switchteste::GetConfig()
{
  if( !switchteste_Command::GetConfig() )
    return false;
//<-dceag-getconfig-e->

  return true;
}

//<-dceag-reg-b->
bool switchteste::Register()
```

```
//<-dceag-reg-e->
{
  return Connect(PK_DeviceTemplate_get());
}

//<-dceag-createinst-b->
switchteste_Command *Create_switchteste(Command_Impl *pPrimaryDeviceCommand,
    DeviceData_Impl *pData, Event_Impl *pEvent, Router *pRouter)
{
  return new switchteste(pPrimaryDeviceCommand, pData, pEvent, pRouter);
}
//<-dceag-createinst-e->


//<-dceag-cmdch-b->
void switchteste::ReceivedCommandForChild(DeviceData_Impl *pDeviceData_Impl,
   string &sCMD_Result,Message *pMessage)
//<-dceag-cmdch-e->
{


  sCMD_Result = "UNHANDLED CHILD";
}


//<-dceag-cmduk-b->
void switchteste::ReceivedUnknownCommand(string &sCMD_Result,Message *
   pMessage)
//<-dceag-cmduk-e->
{
  sCMD_Result = "UNKNOWN COMMAND";
}

void switchteste::Get_State(int iPK_Pipe,string sPK_Device_Pipes,string &
   sCMD_Result,Message *pMessage)
//<-dceag-c192-e->

{


//parte temporal
FILE *fp;
if((fp = fopen("temposmedidos", "a"))==NULL) {
  printf("Cannot open file.\n");
  exit(1);
}


    struct timeval t1, t2;
    double elapsedTime;
  char tempo[20] = { '\0' };
  //fim
```

```
  int sockfd, portno, nbytes;
struct sockaddr_in serv_addr;
struct hostent *server;
char buffersend[256], bufferrecv[256;
string ip_addr,id;


  portno = GetPort();  //lê valor da porta do sensor a partir do template

sockfd = socket(AF_INET, SOCK_STREAM, 0);  //abre o socket
if (sockfd < 0) cout << "ERROR opening socket" << endl;

ip_addr = GetIpAddress(); //lê o IP do sensor indicado no template do
    device
id = GetID(); //lê o ID do sensor indicado no template do device


if ((server = gethostbyname(ip_addr.c_str())) == NULL) { //adquire
    hostname
  fprintf(stderr,"ERROR, no such host\n");
  exit(0);
}

//Define parametros dos sockets
bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr,server->
    h_length);
serv_addr.sin_port = htons(portno);


/* Start timer */
gettimeofday(&t1, NULL);



if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
    {  //efectua ligação
  fprintf(stderr,"ERROR connecting\n");
  exit(0);
}


//Envia Request de presença do ID deste dispositivo
bzero(buffersend,256);
strcpy(buffersend,id);
if(send(sockfd,buffersend,strlen(buffersend),0) == -1) {
  fprintf(stderr,"ERROR writing to socket\n");
  exit(0);
}
```

```
//Recebe Confirmação se o ID existe
bzero(bufferrecv,256);
if ((nbytes = recv(sockfd, bufferrecv, MAXDATASIZE-1, 0)) == -1) {
  fprintf(stderr,"ERROR reading from socket\n");
  exit(0);
}

cout << bufferrecv << endl;

if (strcmp(bufferrecv,"ID Fail")) {
  fprintf(stderr,"ERROR, no such ID\n");
  exit(0);
}

//Definicao do comando a enviar
bzero(buffersend,256);
strcpy(buffersend,"GetState");
if(send(sockfd,buffersend,strlen(buffer),0) == -1) {  //envio do comando
  fprintf(stderr,"ERROR writing to socket\n");
  exit(0);
}




bzero(bufferrecv,256);
if ((nbytes = recv(sockfd, bufferrecv, MAXDATASIZE-1, 0)) == -1) {  //
    recepção do resultado do comando
  fprintf(stderr,"ERROR reading from socket\n");
  exit(0);
}




//tratamento do resultado do comando e comunicação ao LinuxMCE
if (strcmp(bufferrecv, "ON")==0){

  printf("Esta Ligado\n");
  m_pEvent->SendMessage( new Message(m_dwPK_Device,
     DEVICETEMPLATE_VirtDev_All_Orbiters_CONST, //envio de popup para o
     LinuxMCE
                    PRIORITY_NORMAL,
                    MESSAGETYPE_COMMAND,
                    COMMAND_Display_Alert_CONST,
                    4,
                    COMMANDPARAMETER_Text_CONST, "Sensor Ligado",
                    COMMANDPARAMETER_Tokens_CONST, "switchteste",
```

```
                        COMMANDPARAMETER_Timeout_CONST, "10",
                        COMMANDPARAMETER_Interruption_CONST, "0")
                );
    }

    else if (strcmp(bufferrecv, "OFF")==0){

      printf("Esta Desligado\n");
      m_pEvent->SendMessage( new Message(m_dwPK_Device,
          DEVICETEMPLATE_VirtDev_All_Orbiters_CONST,  //envio de popup para o
          LinuxMCE
                        PRIORITY_NORMAL,
                        MESSAGETYPE_COMMAND,
                        COMMAND_Display_Alert_CONST,
                        4,
                        COMMANDPARAMETER_Text_CONST, "Sensor Desligado",
                        COMMANDPARAMETER_Tokens_CONST, "switchteste",
                        COMMANDPARAMETER_Timeout_CONST, "10",
                        COMMANDPARAMETER_Interruption_CONST, "0")
                );
    }


    /* Stop timer */
    gettimeofday(&t2, NULL);
    elapsedTime = (t2.tv_sec - t1.tv_sec) * 1000.0;      // sec to ms
      elapsedTime += (t2.tv_usec - t1.tv_usec) / 1000.0;   // us to ms



    sprintf( tempo, "%.3f", elapsedTime );
    strcat(tempo,"\n");
    printf("Run time: %s\n", tempo);

    fputs(tempo,fp);
    fclose(fp);


    close(sockfd);

}
```

B.2 Servidor

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
```

```
#define MAXDATASIZE 256
#define NSENSORES 5

typedef struct {
  int id;
  int state;
  int dyn;  //valor variavel teste
  char info[20]; //tipo de sensor, luminosidade, motion sensor, switch, etc
      ou outra info, apenas para testar strings
}device;


void architecture(int);


void error(char *msg)
{
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
  int sockfd, newsockfd, portno, clilen, pid;
  struct sockaddr_in serv_addr, cli_addr;


  if (argc < 2) {
    fprintf(stderr,"ERROR, no port provided\n");
    exit(1);
  }
  sockfd = socket(AF_INET, SOCK_STREAM, 0);

  signal(SIGCHLD,SIG_IGN); //evitar zombies
  if (sockfd < 0) error("ERROR opening socket");

  //Definição de parâmetros do servidor
  bzero((char *) &serv_addr, sizeof(serv_addr));
  portno = atoi(argv[1]);
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = INADDR_ANY;
  serv_addr.sin_port = htons(portno);

  //Atribuição do socket ao servidor
  if (bind(sockfd, (struct sockaddr *) &serv_addr,
       sizeof(serv_addr)) < 0)
    error("ERROR on binding");
  listen(sockfd,5);
  clilen = sizeof(cli_addr);
```

```
  //Ciclo da função architecture;
  printf("Aguardando Comando \n");
  while (1) {
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
      error("ERROR on accept");
    pid = fork();
    if (pid < 0)
      error("ERROR on fork");
    if (pid == 0)  {
      close(sockfd);
      architecture(newsockfd);
      exit(0);
    }
    else close(newsockfd);
  }
  return 0;
}



void architecture (int sock)
{

  int nbytes,n,id,check=0;
  char buffersend[256],bufferrecv[256];

  device dev[NSENSORES];

  srand ( time(NULL) ); //gerar seed para o rand()


// Inicializacao Teste ////////////////////
  dev[0].id=0;
  dev[1].id=1;
  dev[2].id=2;
  dev[3].id=3;
  dev[4].id=4;

  strcpy(dev[0].info,"switch");
  strcpy(dev[1].info,"motion");
  strcpy(dev[2].info,"luminosidade");
  strcpy(dev[3].info,"som");
  strcpy(dev[4].info,"switch");

  for (n=0; n<NSENSORES; n++) {
    dev[n].dyn=rand() % 100;
    dev[n].state=rand() % 2;
  }

//////////////////////////////////////////
```

```
//Recepção do ID

bzero(bufferrecv,256);

if ((nbytes = recv(sock, bufferrecv, MAXDATASIZE-1, 0)) == -1) error("
   ERROR reading from socket");

printf("\n\rID = %s\n\r",bufferrecv);
id=atoi(bufferrecv);



//Confirmacao de existencia do sensor (e correspondentemente de ID) e
    envio do ACK
bzero(buffersend,256);
for (n=0; n<NSENSORES; n++) {
  if (id==dev[n].id){
    strcpy(buffersend,"ID existe, aguardando comando"); //ou pode
        responder com o campo info, com o tipo de sensor
    if(send(sock,buffersend,strlen(buffersend),0) == -1) error("ERROR
        writing to socket");
    check=1;
  }
  else if (n==NSENSORES-1 && check==0) {
    strcpy(buffersend,"ID Fail");
    printf("ID nao existe\n\r");
    if(send(sock,buffersend,strlen(buffersend),0) == -1) error("ERROR
        writing to socket");
  }
}




//Recepcao de Comando e envio de Resultado

bzero(bufferrecv,256);

if ((nbytes = recv(sock, bufferrecv, MAXDATASIZE-1, 0)) == -1) error("
   ERROR reading from socket");

printf("Recebido o Comando: %s\n\r",bufferrecv);


bzero(buffersend,256);

if(strcmp(bufferrecv,"GetState")==0){
```

```
    printf("Estado do Device: %d\n\r",dev[id].state);
    if (dev[id].state==1) strcpy(buffersend,"ON");
    else if (dev[id].state==0) strcpy(buffersend,"OFF");
    //ou enviar logo o estado, snprintf(buffersend, sizeof(buffersend), "%d
        ", dev[id].state); //mesmo que itoa
    if(send(sock,buffersend,strlen(buffersend),0) == -1) error("ERROR
        writing to socket");
    printf("%s\n\r",buffersend);

  }


  else if(strcmp(bufferrecv,"GetInfo")==0){

    strcpy(buffersend,dev[id].info);
    if(send(sock,buffersend,strlen(buffersend),0) == -1) error("ERROR
        writing to socket");
    printf("%s\n\r",buffersend);

  }


  else if(strcmp(bufferrecv,"GetDyn")==0){

    snprintf(buffersend, sizeof(buffersend), "%d", dev[id].dyn);
    if(send(sock,buffersend,strlen(buffersend),0) == -1) error("ERROR
        writing to socket");
    printf("%s\n\r",buffersend);

  }



  printf("Comando Enviado \n\r");



}
```

Apêndice C

Placa OpenRB

A placa da *OpenRB* utilizada como *switch* no demonstrador foi o modelo *WP18* (figura C.1), que tem as seguintes características:

- Processador: *Intel XScale IXP425 Operating at 533MHz*
- Memória RAM: *64MB SDRAM (Up to 128MB max.)*
- Memória Flash: *8MB FLASH (Up to 32MB max.)*
- Ligações disponíveis:
  - *2 x Type III Mini-PCI Slots*
  - *4 x 10/100 Base-TX Ethernet Ports (with Auto MDI/MDIX)*
  - *1 x 10/100 Base-TX WAN Port*
  - *1 x RS232 Serial Port*

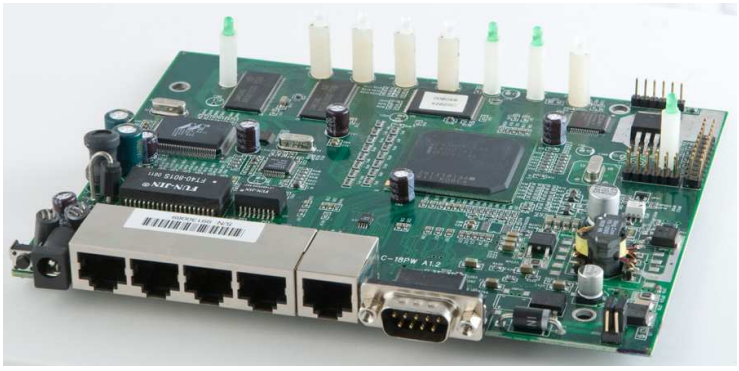

Figura C.1: Placa *OpenRB* WP18 [79]

A nível de configuração, foi apenas necessário desactivar o servidor DHCP, como foi referido anteriormente, acedendo ao *firmware OpenWrt* presente na placa.

Como método de acesso foi usado o SSH, executando num terminal o comando,

```
ssh root@ip
```

e introduzida a password "domotica".

De seguida, com os comandos abaixo apresentados, desactivou-se o DHCP.

```
uci set dhcp.lan.ignore=1
uci commit dhcp
/etc/init.d/dnsmasq restart
```

Assim a placa ficou pronta a funcionar em conjunto com o *LinuxMCE*.

Apêndice D

Kit Powerline

O *Kit Powerline* da Asoka usado no âmbito deste projecto foi o modelo PL9660-ETH (figura D.1), que possui as características descritas de seguida:

- Protocolos de rede suportados:
  - *TCP/IP*
  - *CSMA/CA*
  - *TDMA*
- Interfaces de rede: *Ethernet RJ-45 (1 10/100M port), PLC*
  - *Ethernet RJ-45 (1 10/100M port)*
  - *PLC*
- Largura de banda: *PLC 200 Mbps*
- Segurança incorporada na *Powerline*: *128-bit AES Encryption*
- Frequência de trabalho da*Powerline*: *2-30 MHz*
- Tipos de modulação suportados:
  - *OFDM*
  - *QAM 1024/256/64/16*
  - *QPSK*
  - *BPSK*
- Qualidade de serviço:
  - *ToS*

Figura D.1: Kit *Powerline* PL9660-ETH [80]

- *802.1q*
- *TDMA*

O *kit* é formado por uma caixa branca com uma ficha de corrente, uma porta de *Ethernet* e possui também um botão para restituir o *firmware* de origem e outro para fazer o mesmo com o Network ID.

Nesta tese, foram usados dois *kits* destes, para permitir estabelecer uma ligação ponto-a-ponto entre o *switch* e o AP.

Quanto à configuração, existem algumas opções que podem ser personalizadas através do programa de gestão de rede incluído (*PlugLink AV Power Manager*), tais como, a modificação do nome dos dispositivos e da palavra de segurança, mas não foi necessário alterá-las. Assim, bastou apenas conectar os dois *kits* às tomadas da corrente, e do outro lado, os cabos de rede aos dispositivos pretendidos.

Com o *software* disponibilizado, executaram-se ainda alguns testes qualitativos, que demonstraram que o produto atingia velocidades bastante boas dentro da rede (4 a 6 MB/s), e com latência relativamente baixa, podendo assim ser usado neste trabalho sem causar *bottlenecks*.

Apêndice E

Lista de Acrónimos

A/V Audio/Video

AP Access Point

BIOS Basic Input/Output System

CD Compact Disc

DCE Data Commands Events

DHCP Dynamic Host Configuration Protocol

DNS Domain Name Service

DTS Digital Theater Systems

EDR Enhanced Data Rate

FFT Fast Fourier Transform

GUI Graphical User Interface

HS High Speed

IP Internet Protocol

IRC Internet Relay Chat

ITU International Telecommunication Union

LAN Local Area Network

MAC Medium Access Control

MD Media Director

NIC Network Interface Controller

OFDM Orthogonal Frequency-Division Multiplexing

OSI Open Systems Interconnection

PC Personal Computer

PDA Personal Digital Assistants

QAM Quadrature Amplitude Modulation

QoS Quality of Service

RC Release Candidate

RF Radio Frequency

SQL Structured Query Language

SSH Secure SHell

SSL Secure Sockets Layer

SVN SubVersioN

TDMA Time Division Multiple Access

URL Uniform Resource Locator

USB Universal Serial Bus

VoIP Voice over Internet Protocol

WEP Wireless Encryption Protocol

Wi-FI Wireless Fidelity

WPA Wi-Fi Protected Access

WPA2 Wi-Fi Protected Access 2

WPAN Wireless Personal Area Network

ZC Zigbee Coordinator

ZR Zigbee Router

ZED ZigBee End Device

Bibliografia

[1] Alves, J. A. e Mota, J. *Casas Inteligentes*. Edições Centro Atlântico, 2003.

[2] Cisco Systems. Internetworking technology handbook [online]. Available from: `http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Ethernet.html` [cited 2010-05-15].

[3] IEEE 802.3 Ethernet Working Group. Ieee 802.3 ethernet [online]. Available from: `http://grouper.ieee.org/groups/802/3/index.html` [cited 2010-05-15].

[4] IEEE P802.3 Task Force. Ieee 802.3 ethernet poe [online]. Available from: `http://grouper.ieee.org/groups/802/3/at/index.html` [cited 2010-05-15].

[5] IEEE P802.3ba Task Force. Ieee p802.3ba 40gb/s and 100gb/s ethernet task force [online]. Available from: `http://www.ieee802.org/3/ba/` [cited 2010-07-22].

[6] NetworkWorld. 40gbps and 100gbps ethernet in 2016 [online]. Available from: `http://www.networkworld.com/newsletters/lans/2008/0211lan2.html` [cited 2010-07-22].

[7] Bluetooth SIG. Bluetooth basics [online]. Available from: `http://www.bluetooth.com/English/Technology/Pages/Basics.aspx` [cited 2010-05-15].

[8] Padgette, J. e Scarfone, K. Guide to bluetooth security, recommendations of the national institute of standards and technology. *NIST Special Publication 800-121*, pages 2–1, 2–3, 2–4, September 2008.

[9] Bluetooth SIG. Bluetooth 2.1 edr specifications [online]. Available from: `http://www.bluetooth.com/English/Technology/Works/Pages/Core_Specification_v21__EDR.aspx` [cited 2010-05-15].

[10] Bluetooth SIG. Bluetooth 3.0 hs specifications [online]. Available from: `http://www.bluetooth.com/English/Technology/Works/Pages/Core_Specification_v30.aspx` [cited 2010-05-15].

[11] Bluetooth SIG. Bluetooth 4.0 lp specifications [online]. Available from: `http://www.bluetooth.com/English/Technology/Works/Pages/Bluetooth_low_energy_technology.aspx` [cited 2010-05-15].

[12] Bluetooth SIG. Bluetooth 4.0 specification finished [online]. Available from: http://www.bluetooth.com/English/Press/Pages/PressReleasesDetail.aspx?ID=101 [cited 2010-05-15].

[13] TechRepublic. Topologia do bluetooth [online]. Available from: http://articles.techrepublic.com.com/5100-10878_11-6139987.html [cited 2010-05-24].

[14] Zigbee Alliance. Sítio online da zigbee alliance [online]. Available from: http://www.zigbee.org/ [cited 2010-05-15].

[15] Zigbee Alliance. Zigbee technical documents [online]. Available from: http://www.zigbee.org/Products/DownloadZigBeeTechnicalDocuments.aspx [cited 2010-05-15].

[16] CommsDesign. Arquitectura do zigbee a nível de rede [online]. Available from: http://www.commsdesign.com/showArticle.jhtml?articleID=192200323 [cited 2010-05-24].

[17] ITU. Sítio online da itu [online]. Available from: http://www.itu.int/ [cited 2010-05-15].

[18] Intel. Helping unify the home networks market by accelerating development and acceptance of the itu-t g.hn standard. *Intel and The HomeGrid Forum*, 2009. Available from: http://www.intel.com/standards/case/HomeGrid_Case_Study.pdf.

[19] Everywire. Att participating in g.hn standard development [online]. Available from: http://www.everywire.com/2009/02/att-participating-in-ghn-standard-development-1.html [cited 2010-05-15].

[20] HomeGrid Forum. Bt joins board of directors [online]. Available from: http://www.homegridforum.org/news_events/pr/05_27_09/ [cited 2010-05-15].

[21] HomeGrid Forum. Industry creates homegrid forum to develop technology for enjoying multimedia anywhere in the home [online]. Available from: http://www.homegridforum.org/news_events/pr/04_29_08/ [cited 2010-05-15].

[22] Sean Buckley. Homegrid and best buy - a logical marriage [online]. Available from: http://www.telecomengine.com/techzones/wireless/article.asp?HH_ID=AR_5000 [cited 2010-05-15].

[23] HomeGrid Forum. Homegrid ghn resource library [online]. Available from: http://www.homegridforum.org/resource_library [cited 2010-05-15].

[24] DS2 Blog. Top ten things you need to know about the new g.hn standard [online]. Available from: http://blog.ds2.es/ds2blog/2009/05/top-ten-things-about-ghn-standard.html [cited 2010-05-15].

[25] DS2 Blog. How fast can g.hn be? [online]. Available from: http://blog.ds2.es/ds2blog/2009/04/how-fast.html [cited 2010-05-15].

[26] EuroX10. X10info [online]. Available from: http://www.eurox10.com/Content/X10Information.htm [cited 2010-05-15].

[27] Silva, Luís F. G. Automação em ambientes residenciais. Master's thesis, Universidade de Aveiro, 2008.

[28] Inc. SmartLabs. Sítio online da smartlabs [online]. Available from: http://www.smartlabsinc.com/ [cited 2010-05-15].

[29] Smarthome Technology. Insteon the details, page 4 [online]. Available from: http://www.insteon.net/pdf/Insteondetails.pdf [cited 2010-05-15].

[30] Electronic Design. Topologia do insteon [online]. Available from: http://electronicdesign.com/article/boards-modules-systems/refresh-insteon-technology33281.aspx [cited 2010-05-24].

[31] Z-WaveAlliance. Sítio online da z-wave alliance [online]. Available from: http://www.z-wavealliance.org [cited 2010-05-15].

[32] Z-WaveAlliance. Z-wave benefits [online]. Available from: http://www.z-wavealliance.org/modules/Benefits/ [cited 2010-05-15].

[33] Echelon Corporation. Lonworks technology overview [online]. Available from: http://www.echelon.com/communities/energycontrol/developers//LonWorks/default.htm [cited 2010-05-15].

[34] Echelon Corporation. Sítio online da echelon [online]. Available from: http://www.echelon.com/ [cited 2010-05-15].

[35] Echelon Corporation. Neuron chips [online]. Available from: http://www.echelon.com/communities/energycontrol/developers//LonWorks/neuron.htm [cited 2010-05-15].

[36] Echelon Corporation. Topologia do lonworks [online]. Available from: http://www.echelon.com/communities/energycontrol/developers//lonworks/default.htm [cited 2010-05-24].

[37] KNX Association. Knx main advantages [online]. Available from: http://www.knx.org/knx-standard/main-advantages/ [cited 2010-05-15].

[38] KNX Association. Sítio online do knx [online]. Available from: http://www.knx.org/ [cited 2010-05-15].

[39] Creston Electronics. Sítio online da creston [online]. Available from: http://www.crestron.com/ [cited 2010-05-24].

[40] AMX. Sítio online da amx [online]. Available from: http://www.amx.com/ [cited 2010-05-24].

[41] HomeSeer. Sítio online da homeseer [online]. Available from: http://www.homeseer.com/ [cited 2010-05-24].

[42] Nathan Willis. Home automation with linux [online]. Available from: http://www.linux.com/news/hardware/peripherals/135780-home-automation-with-linux [cited 2010-05-24].

[43] Microsoft. Sítio online do windows media center [online]. Available from: http://www.microsoft.com/windows/windows-media-center/get-started/default.aspx [cited 2010-05-24].

[44] MisterHouse. Sítio online do misterhouse [online]. Available from: http://misterhouse.sourceforge.net/ [cited 2010-05-24].

[45] OpenRemote. Sítio online do openremote [online]. Available from: http://www.openremote.org/display/HOME/OpenRemote.

[46] LinuxMCE. Sítio online do linuxmce [online]. Available from: http://www.linuxmce.org/ [cited 2010-05-15].

[47] Pluto. Sítio online do pluto [online]. Available from: http://www.plutohome.com [cited 2010-05-15].

[48] Xine. Sítio online do xine [online]. Available from: http://www.xine-project.org/ [cited 2010-05-15].

[49] Asterisk. Sítio online do asterisk [online]. Available from: http://www.asterisk.org/ [cited 2010-05-15].

[50] MythTv. Sítio online da mythtv [online]. Available from: http://www.mythtv.org/ [cited 2010-05-15].

[51] LinuxMCE. Sítio online da svn do linuxmce [online]. Available from: http://svn.linuxmce.org/svn/trunk/ [cited 2010-05-15].

[52] LinuxMCE. Sítio online da wiki do linuxmce [online]. Available from: http://wiki.linuxmce.org [cited 2010-05-15].

[53] LinuxMCE. Imagens do linuxmce [online]. Available from: http://linuxmce.org/index.php/gallery/screenshots [cited 2010-05-24].

[54] LinuxMCE. Parte de multimédia do linuxmce [online]. Available from: http://linuxmce.org/index.php/features/media [cited 2010-05-24].

[55] LinuxMCE. Parte de telecomunicações do linuxmce [online]. Available from: http://linuxmce.org/index.php/features/telecom [cited 2010-05-24].

[56] LinuxMCE. Parte de segurança do linuxmce [online]. Available from: http://linuxmce.org/index.php/features/security [cited 2010-05-24].

[57] LinuxMCE. Parte de iluminação do linuxmce [online]. Available from: http://linuxmce.org/index.php/features/lighting [cited 2010-05-24].

[58] LinuxMCE. Parte de climatização do linuxmce [online]. Available from: http://linuxmce.org/index.php/features/climate [cited 2010-05-24].

[59] LinuxMCE. Diagrama da configuração do linuxmce com core pc dedicado [online]. Available from: http://wiki.linuxmce.org/index.php/Image:Diagram1.jpg [cited 2010-05-24].

[60] LinuxMCE. Diagrama da configuração do linuxmce com core pc híbrido [online]. Available from: http://wiki.linuxmce.org/index.php/Image:Diagram2.jpg [cited 2010-05-24].

[61] LinuxMCE. Arquitectura do linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:LMCE_Architecture_mini.jpg [cited 2010-05-24].

[62] LinuxMCE. Estrutura do dcerouter no linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:DCERouter.jpg [cited 2010-05-24].

[63] OpenWrt. Sítio online do openwrt [online]. Available from: http://openwrt.org/ [cited 2010-05-15].

[64] OpenWrt. Packages para o openwrt [online]. Available from: http://downloads.openwrt.org/whiterussian/packages/ [cited 2010-05-15].

[65] Calvert, K. L. e Donahoo, M. J. *The pocket guide to TCP/IP sockets : C version*. Morgan Kaufmann, 2001.

[66] Calvert, K. L. e Donahoo, M. J. *TCP/IP Sockets in C, Second Edition: Practical Guide for Programmers*. Morgan Kaufmann, 2009.

[67] OpenRB. Sítio online da openrb [online]. Available from: http://shop.openrb.com/index.php?main_page=index [cited 2010-05-24].

[68] Asoka. Sítio online da asoka [online]. Available from: http://www.asokausa.com/ [cited 2010-05-24].

[69] Verisign. Sítio online da verisign [online]. Available from: http://www.verisign.com/ [cited 2010-05-24].

[70] LinuxMCE. Ecrã inicial do a/v wizard no linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:WelcomeAV.jpg [cited 2010-05-24].

[71] LinuxMCE. Ecrã de escolha das resoluções no a/v wizard do linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:Avw1.jpg [cited 2010-05-24].

[72] LinuxMCE. Ecrã de escolha do aspecto gráfico no a/v wizard do linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:Avw8.jpg [cited 2010-05-24].

[73] LinuxMCE. Ecrã de selecção da ligação de som no a/v wizard do linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:Avw10.jpg [cited 2010-05-24].

[74] LinuxMCE. Ecrã de resumo final do a/v wizard no linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:Avw12.jpg [cited 2010-05-24].

[75] LinuxMCE. Ecrã inicial do house wizard no linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:SeeAndHearMe.jpg [cited 2010-05-24].

[76] LinuxMCE. Ecrã de definição dos utilizadores da casa no house wizard do linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:UsersWizard.jpg [cited 2010-05-24].

[77] LinuxMCE. Ecrã de selecção das divisões da casa no house wizard do linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:RoomsHouseWizard.jpg [cited 2010-05-24].

[78] LinuxMCE. Ecrã do resumo final do house wizard no linuxmce [online]. Available from: http://wiki.linuxmce.org/index.php/Image:DoneHouseWizard.jpg [cited 2010-05-24].

[79] OpenRB. Placa openrb wp18 [online]. Available from: http://shop.openrb.com/index.php?main_page=product_info&cPath=18&products_id=116 [cited 2010-05-24].

[80] Asoka. Kit powerline pl9660-eth [online]. Available from: http://www.asokashop.eu/index.php/en/products-mainmenu-64/pluglink9660eth.html [cited 2010-05-24].